Infinitary languages and fully abstract models of fair asynchrony

Philippe Darondeau

IRIA

CENTRE DE RENNES
IRISA

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél.(3) 954 90 20

Rapports de Recherche

N° 330

INFINITARY LANGUAGES AND FULLY ABSTRACT MODELS OF FAIR ASYNCHRONY

Philippe DARONDEAU

Septembre 1984

IRISA

INSTITUT DE RECHERCHE EN INFORMATIQUE
ET SYSTÈMES ALÉATOIRES

Campus Universitaire de Beaulieu
Avenue du Général Leclerc
35042 - RENNES CÉDEX

INFINITARY LANGUAGES AND FULLY ABSTRACT MODELS OF FAIR ASYNCHRONY

Ph. Darondeau

CNRS - IRISA

Publication Interne n° 230
Juillet 1984
79 pages

Abstract Assuming strong fairness for a CCS like language, we construct a fully abstract model of the implementation preorder : $p \tilde{\sqsubseteq} q$ iff for every context $\mathcal{C}$ and for every program r, the computation of r in $(\mathcal{C}(p) \mid r)$ does not allow to recognize that q has been replaced by p.

Résumé Supposant l'équité forte pour un langage à la CCS, nous construisons un modèle pleinement abstrait du préordre d'implémentation : $p \tilde{\sqsubseteq} q$ ssi pour tout contexte $\mathcal{C}$ et pour tout programme r, le calcul de r dans $(\mathcal{C}(p) \mid r)$ ne permet pas de reconnaître que q a été remplacé par p.

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE
(L.A. 227)
UNIVERSITÉ DE RENNES 1 I.N.S.A. DE RENNES

INSTITUT NATIONAL DE RECHERCHE
EN INFORMATIQUE ET EN AUTOMATIQUE
(LABORATOIRE DE RENNES)

CONTENTS

1. INTRODUCTION

Programs considered here are asynchronous programs in the style of pure CCS without value passing [Mi1]. Accordingly, no provision will be made for simultaneous actions / events, taken instead as a basic phenomenon in synchronous calculi such as SCCS [Mi2]. Two subsets of pure CCS, namely the "finite" and the "rational" subsets, have already been studied by the author with the intend of finding out complete proof systems for the observational equivalence and congruence of programs. Such studies involve several categories of problems :

1 - how to observe the behaviour of a program, and what relations should be used to compare behaviours
2 - how to encode sets of observations into a domain
3 - how to construct a model in which two programs are identified iff in no context they can be observed different
4 - how to obtain sound and complete proof systems for such "fully abstract" models in the sense of [Mi3].

Problems 3 and 4 may also be raised with a look at observational preorders instead of equivalences, as was first attempted in [HP].

For finite behaviour programs, the difficulties lay in problem 1 : once answers have been devised, a complete proof system may be found in the form of a finite family of (in-) equational axiom schemas, whose initial model solves problems 2 and 3 [HM]. If the observational equivalence is chosen recursive w.r.t. the unrolling tree of a transition system, it is implicitly assumed that an observer is at each time able to obtain copies of the observed system in its current state or to command backtracking. Thus, programs are not observed in their ordinary working conditions and the equivalence is too discriminative (that criticism may be addressed as well to the bisimulation equivalences of SCCS and MEIJE [AB]). More appropriate is the concept of an "on line" observer which interacts with the observed program in linear runs from the initial state. Finite deterministic observers of that type have been suggested in [HBR]. In that work, the meaning of a program is a set of failures $\langle \rho, \delta \rangle$, where $\rho \in Actions^*$ is a sequence of interactions with the observer, and $\delta \subseteq Actions$ is a refusal, that is an interaction attempt which may remain forever unsatisfied after ρ. In two different settings, the resulting equivalence has been axiomatized in [Da] and in [Br1]. More precisely,

the last study gives a formalization of the preorder $\tilde{\sqsubseteq}$ obtained by reversing the inclusion of failure sets. To some extent, the above proof systems agree with the axiomatization constructed by Hennessy and de Nicola for their testing preorders [HN]. There, the on line observers become more varied and more substantial : they are identified with possibly nondeterministic or parallel programs which realize tests in the sense of Kennaway [K]. The binary result of a test abstracts a maximal computation of the system (testee | tester). Program p is improved by program q iff for any test t which may (must) be passed by p, q may (must) also pass t. This preorder is in fact generated by the subset of the deterministic observers, and $\tilde{\sqsubseteq}$ is recovered when may is deleted [Br2]. Associated to the improvement preorder is the domain of node trees (NT) studied in [He1]. Given a close node tree, the language attached to its paths may be seen as a set of pairs $\langle p, \delta \rangle$, where the acceptance set δ indicates some fictitious state reachable by p (but open trees are also available for representing partially defined programs). No corresponding domain exists for the implementation preorder : p implements q iff for any test t which may (must) be passed

by $p(q)$, $q(p)$ may (must) also pass t.

Let us now turn more specifically to recursively defined programs. Since the syntactic approximation agrees with the preorders considered so far, standard completion techniques may be applied to cope with the infinitary case, wether refusal sets or acceptance trees are preferred. Using these completion techniques, which assume the ω-continuity of the operations on processes, recursive programs are supplied with semantics given as infinite sets of finitary structures. This feature involves infinitary rules of induction in the associated proof systems [HN] [Br2], but more severely, it also entails the lack of fairness [DK2]. In the context of CCS, the strong fairness assumption which makes sense should be expressed as: "no agent who infinitely often has opportunities to communicate with other agents remains forever inactive", which strongly departs from the classical finite delay property explicit in [Mi4] and implicit in [CS]. Not every parallel process has a sequential nondeterministic equivalent! A first attempt was made in [DK1] to embody fairness into an observational model of asynchronous CCS. Seeking in that work for the greatest subset of CCS which preserves decidability

of the fair observation equivalence, we found that this requirement fits exactly with the restriction of bounded parallelism. We then defined a model relating each program with a rational language of triples $\langle \delta, \rho, d \rangle$, which look like extended refusals. The informal meaning is as follows : $\rho \in \text{Actions}^* \cup \text{Actions}^\omega$ is a finite or infinite sequence of interactions with a finite family of communicating observers, $\delta \subseteq \text{Actions}$ is the set union of the observers' failures, and $d \subseteq \text{Actions}$ is the set of their infinitely often repeated offers. (Notice that the observation $\langle \emptyset, \alpha^\omega, \{\alpha\} \rangle$ costs no more time than the observation $\langle \{\alpha\}, \in, \emptyset \rangle$!). Here, the semantics of a program is an infinite set of infinitary structures. Owing to the confinement within rational languages, and despite the weak continuity, this feature allows for a complete proof system without induction. Unhappily, the observation equivalence of [DK1] was not defined in a strictly operational way. No such limitation appears in the testing model VENT constructed by Hennessy for synchronous CCS with a weakly fair delay operator [He 2]. There, the semantics of programs are given by least fixed points of monotonic (weakly continuous) functions. The objects of the model are infinitary acceptance trees with additional "limit points". Among these,

close trees may also be seen as sets of finitary acceptance pairs $\langle \rho, \delta \rangle$ together with infinitary traces $\rho \in Actions^{\omega}$. But once more, open trees are also available. Full abstractness of the model has been established for terms without free variables, i.e. for programs, but no such property holds for the open terms.

The present work draws its inspiration from all the above: using as a semantic domain the infinitary languages of triples $\langle \delta, \rho, d \rangle$ already called for in [DK1], we try to obtain for asynchrony and strong fairness an analogue of the results established in [HE2] for synchrony and weak fairness. For a particular subset of CCS, we will construct a fully abstract model of the following preorder [JO]: p implements q iff. for every context C and for every program r, the computations of r in $(C(p) \mid r)$ do not allow to recognize that q has been replaced by p. The improvement ordering of Hennessy and de Nicola has no counterpart here, due to the infinite duration of the observations: if p does not implement q, then for some r there exists some computation which is possible in $(p \mid r)$ and not in $(q \mid r)$! Remark that in the case of tests, may and must are also diffe-

rentiated by computations with infinite duration .

To make things quite clear , we have to precise which subset of CCS is dealt with . For the unrestricted version of pure CCS , we did not succeed in finding out a model whose full abstraction property is valid for every open terms . Although we are not able to explain the exact reason of that failure , we feel that it is inherent to the consideration of fairness. But our initial intend was to reach uniform full abstractness, and we therefore attempted to turn pure CCS , or at least a non trivial subset of it , into a conventional Σ-algebra . The language studied below is the result of this attempt . The full power of Turing machines is reached by means of rudimentary network iterators . More elaborate forms of iterators may be worth consideration in the future .

The remainder of the text is organized as follows . Section 2 introduces the language and its operational definition . Section 3 is devoted to the construction of an operational model . Section 4 derives from the above a fully observational model . Section 5 gives further details for the rational subset . Section 6 draws some conclusions .

2. THE LANGUAGE

2.1 THE SYNTACTIC DEFINITION

The language studied here, call it CS, evolved from the pure version of the asynchronous CCS [ ]. The main alterations are as follows. Polyadic guarding operators replace the sum operator, hence no confusion exists between synchronous and asynchronous alternatives. Recursive definitions have been obliterated: in CS, every term of the language is a fully defined program. The limitations due to the latter feature are balanced by the supply of iterators for defining cyclic agents and unbounded strings of communicating agents, which may be used in the same way as in pure CCS to simulate Turing machines.

In the sequel, we assume given disjoint sets of complementary <u>action names</u> Δ and $\overline{\Delta}$, ranged over by α resp. $\overline{\alpha}$, such that there exist reciprocal bijections $\alpha \longrightarrow \overline{\alpha} \longrightarrow \alpha$. We let <u>Ren</u>, ranged over by π, denote the set of domain finite injections over $\Lambda = \Delta \cup \overline{\Delta}$ such that:

$(\forall \lambda)(\pi(\lambda)$ defined $\Rightarrow \pi(\overline{\lambda}) = \overline{\pi(\lambda)}) \;\&$

$(\exists n)(\forall \lambda)(\pi^n(\lambda)$ undefined $)$.

CS is the free term algebra T_Σ for the signature $\Sigma = \bigcup \Sigma_n$, $n \geq 0$ given by :

$\Sigma_0 = \{NIL\} \cup \{ < \lambda \Rightarrow \lambda_1 : \lambda'_1, \ldots, \lambda_n : \lambda'_n > \mid \lambda, \lambda_i, \lambda'_i \in \Lambda \}$

$\Sigma_1 = \{ < \lambda > \mid \lambda \in \Lambda \} \cup \{ [\pi] \mid \pi \in Ren \} \cup \{ [\lambda \triangleright \pi] \mid \lambda \in \Lambda, \pi \in Ren \}$

$\Sigma_2 = \{ | \} \cup \{ < \lambda_1, \lambda_2 > \mid \lambda_i \in \Lambda \}$

$\Sigma_n = \{ < \lambda_1, \ldots, \lambda_n > \mid \lambda_i \in \Lambda \}$ for $n > 2$.

In Σ_2 for instance, $< \lambda_1, \lambda_2 >$ denotes a binary guarding operator and $|$ is the asynchronous composition. Let π be the renaming function with domain $\{\alpha, \bar{\alpha}\}$ such that $\pi(\alpha) = \beta$, then $p[\alpha \triangleright \pi]$ is intuitively the unbounded string $p_1 \frown p_2 \frown \ldots \frown p_i \frown \ldots$ of copies of p each of which, once started from the left by an α, may further communicate with its left neighbour by α's (or $\bar{\alpha}$'s) perceived as β's (or $\bar{\beta}$'s). Thus, $p[\alpha \triangleright \pi]$ might be specified by the recursive statement $X \Leftarrow < \alpha > (p \mid X[\pi])$, but this is not a Σ-term. Due to the definition of Ren, the possible influence of an agent in $p[\alpha \triangleright \pi]$ is confined within an interval of maximal size $2n$ if π^n is the undefined function. Hence, string iterators $[\alpha \triangleright \pi]$ taken alone do not provide sufficient means for programming infinite behaviours of finite automata. Opportunately, such behaviours may be rendered by finite compositions of local iterators $< \lambda \Rightarrow \{\lambda_i : \lambda'_i\}_i >$. A local iterator might be specified by the recursive

statement $X \Leftarrow \langle\lambda\rangle(\langle\lambda_1,..,\lambda_n\rangle(\langle\lambda'_1\rangle(X),..,\langle\lambda'_n\rangle(X)))$, but once more, this is not a Σ-term. The power of CS comes from string iterators applied to finite compositions of local iterators.

Our programs are endowed with <u>sorts</u>, which are symmetric subsets of Λ. The inheritance of sorts is determined by the following rules, where we let $sym(L) = L \cup \bar{L}$ for any subset L of Λ:

$sort(Nil) = \phi$

$sort(\langle\lambda \Rightarrow \lambda_1 : \lambda'_1, \ldots, \lambda_n : \lambda'_n\rangle) =$

$\quad sym(\{\lambda, \lambda_1, \lambda'_1, \ldots, \lambda_n, \lambda'_n\})$

$sort(\langle\lambda_1,\ldots,\lambda_n\rangle(t_1,\ldots,t_n)) =$

$\quad sym(\{\lambda_1,\ldots,\lambda_n\}) \cup (\bigcup_i sort(t_i))$

$sort(t_1 | t_2) = sort(t_1) \cup sort(t_2)$

$sort(t[\pi]) = \pi(sort(t))$

$sort(t[\lambda \triangleright \pi]) = \pi^*(\{\lambda, \bar{\lambda}\} \cup \pi(sort(t)))$

where $\pi^*(E)$ is defined as $\Sigma \pi^n(E)$, $n \geq 0$.

We shall see later that the sort S of an agent is a superset of the set of names by which the agent may interact with its environment. Now, the induction on the structure of terms allows us to state the ...

<u>lemma 2.1</u>. For any t in T_Σ, $sort(t)$ is finite □

2.2. THE OPERATIONAL DEFINITION

In order to supply T_Σ with an operational definition, we make first a rough confusion between parallel programs and transition systems without distinguished states. The elementary transitions are defined axiomatically as labelled reductions of terms. Since the above confusion is undue, special restrictions are then taken to rule out those sequences of transitions which are not the accounts of any fair computation.

A <u>computation</u> is a sequence of <u>communications</u> between the agents of an insulated system, whose global state is described by a program in T_Σ. A communication may be represented by an associated transition $p_{i-1} \xrightarrow{e_i} p_i$ between program states, where e_i is a pair of visible actions $\{\lambda_R, \bar{\lambda}_S\}$ or a pair of invisible actions $\{\tau_R, \tau_S\}$. Subscripts of actions identify sequential <u>processes</u>. Along the computation, fresh identifiers are generated for new processes each time a new unguarded occurence of the composition operator symbol | appears in the program state, which is the case at each expansion of an iterator. Process identifiers are otherwise inherited along the derivation of terms into subterms.

Processes are thus finite sequences of actions. In a <u>strongly fair</u> computation, a process who infinitely often has opportunities to communicate with other processes necessarily does so. Told otherwise, a fair computation of a program p_0 is a finite or infinite sequence of communications $\langle p_{i-1} \xrightarrow{e_i} p_i \rangle_{i \in I}$ such that $(\forall R)(\exists j)(\forall k \geq j)(p_k \xrightarrow{\mu_R / \bar{\mu}_S})$.

Communications and actions of programs / processes may be specified together by the axiomatic system $[\xrightarrow{\bullet}]$ given below, using the following ...

<u>Conventions</u>

- λ ranges over Λ; $\tau = \bar{\tau} \notin \Lambda$ is the undefined action name, thus $\pi(\lambda) = \tau$ if $\lambda \notin \mathrm{dom}(\pi)$ and $\pi(\tau) = \tau$ for any π in Ren; μ ranges over $\Lambda \cup \{\tau\}$.
- $\underline{\mathrm{ID}} = \{\leftarrow, \rightarrow\}^*$ is the set of process identifiers, ranged over by R, S. Arrows have a positional meaning with respect to the asynchronous composition operator $|$.
- $\underline{E}$, ranged over by e, is the union $\underline{E}_1 \cup \underline{E}_2$ of the sets with respective elements $\{\lambda_R\}$ or $\{\mu_R, \bar{\mu}_S\}$; for $T \in \mathrm{ID}$, $T(e)$ denotes accordingly the set $\{\lambda_{TR}\}$ or the set $\{\mu_{TR}, \bar{\mu}_{TS}\}$.

The action rules $[\xrightarrow{\bullet}]$

$$\langle \lambda_1, \ldots, \lambda_n \rangle (t_1, \ldots, t_n) \xrightarrow{\lambda_i} t_i \quad , \quad i \in \{1 \ldots n\} \qquad \text{I}$$

$$\frac{t \xrightarrow{e} t'}{(t \mid u) \xrightarrow{\leftarrow (e)} (t' \mid u)} \qquad \frac{t \xrightarrow{e} t'}{(u \mid t) \xrightarrow{\rightarrow (e)} (u \mid t')} \qquad \text{II}$$

$$\frac{t \xrightarrow{\lambda_R} t' \quad , \quad u \xrightarrow{\bar{\lambda}_S} u'}{(t \mid u) \xrightarrow{\lambda \leftarrow R \,,\, \bar{\lambda} \rightarrow S} (t' \mid u')} \qquad \text{III}$$

$$\frac{t \xrightarrow{\lambda_R} t'}{t[\pi] \xrightarrow{\pi(\lambda)_R} t'[\pi]} \quad \text{if } \pi(\lambda) \neq \tau \qquad \text{IV}$$

$$\frac{t \xrightarrow{\mu_R \; \bar{\mu}_S} t'}{t[\pi] \xrightarrow{\pi(\mu)_R \; \pi(\bar{\mu})_S} t'[\pi]} \qquad \text{V}$$

$$t[\lambda \triangleright \pi] \xrightarrow{\lambda} (t \mid t[\lambda \triangleright \pi])[\pi] \qquad \text{VI}$$

$$\text{if } t = \langle \lambda \Longrightarrow \lambda_1 : \lambda'_1 \,, \ldots, \lambda_n : \lambda'_n \rangle \text{ then} \qquad \text{VII}$$

$$t \xrightarrow{\lambda} (\text{NIL} \mid \langle \lambda_1, \ldots, \lambda_n \rangle (\langle \lambda'_1 \rangle (t), \ldots, \langle \lambda'_n \rangle (t)))$$

The next two lemmas state easy properties of the above system.

<u>lemma 2.2</u> For any transition $t \xrightarrow{e} t'$, $sort(t')$ is a subset of $sort(t)$ and every actions in e have their names in $sort(t) \cup \{\tau\}$ □

<u>lemma 2.3</u> For any $t \in T_\Sigma$, there exist a finite number of transitions $t \xrightarrow{e} t'$ with origin t. □

The operational setting may now be summarized by two definitions.

<u>definition 2.1</u> For $t \in T_\Sigma$ and $R \in iD$, the set $Ext_R(t)$ resp. $Int_R(t)$ of t's <u>external</u> resp. <u>internal</u> <u>R-actions</u> is the set $\{\mu \mid (\exists u)(t \xRightarrow{\mu_R} u)\}$ for $\xRightarrow{\mu_R} = \xrightarrow{\mu_R}$ resp. $\xRightarrow{\mu_R} = \bigcup_S \xrightarrow{\mu_R \bar{\mu}_S}$. Similarly, we let $Ext(t) = \bigcup_R Ext_R(t)$, $Int(t) = \bigcup_R Int_R(t)$.

<u>definition 2.2</u> For p_0 in T_Σ, the associated set <u>fair</u>(p_0) of <u>closed fair computations</u> is the set of sequences $(p_i, e_i)_{i \in \omega}$ such that the following conditions are satisfied, letting $E_0 = \{\emptyset\}$:

1 - $(\forall i)((e_i \in E_2 \;\&\; p_i \xrightarrow{e_i} p_{i+1}) \vee (e_i = \emptyset \;\&\; p_i = p_{i+1}))$

2 - $(e_i)_{i \in \omega} \in (E_2{}^\omega \cup E_2{}^* E_0{}^\omega)$

3 - $(\forall R \in iD)(\exists j)(\forall k \geq j)(Int_R(p_k) = \emptyset)$.

Henceforth, we call a <u>closed computation</u> every sequence $(p_i, e_i)_{i \in \omega}$ which satisfies 1 and 2.

The current division ends with a series of lemmas which state simple facts used in the sequel.

<u>lemma 2.4</u> $p \xrightarrow{\lambda_R} p' \;\&\; p \xrightarrow{\bar{\lambda}_S} p'' \;\&\; R \neq S \Rightarrow p \xrightarrow{\lambda_R \bar{\lambda}_S} q$ for some q.

<u>proof</u> by the injectivity of renaming functions □

<u>lemma 2.5</u> $p \xrightarrow{e} q \;\&\; Ext_R(p) \neq \emptyset \Rightarrow Ext_R(q) = Ext_R(p) \vee (\exists \mu)(\mu_R \in e)$ □

lemma 2.6 For any closed computation $(\mu_i, e_i)_{i\in\omega}$ and for any process identifier $R \in ID$, there exist a finite number of index values i such that $\mu_R \in e_i$ for some μ. □

lemma 2.7 For any closed computation $(\mu_i, e_i)_{i\in\omega}$:

$$\bigsqcup \{ \bigcap_{k\geq i} Ext(\mu_k) \mid i \in \omega \} = \bigsqcap \{ \bigcup_{k\geq i} Ext(\mu_k) \mid i \in \omega \}.$$

proof: if Ext is now replaced by Ext_R, then the equality must be satisfied for every R by lemmas 2.5 and 2.6 □

lemma 2.8 For any μ_0 in T_Σ, fair(μ_0) is not the empty set.

proof Let $\leq$ be the order relation on ID such that $W \leq W'$ iff $((lgth(W) < lgth(W'))$ or $(W = U \leftarrow V \;\&\; W' = U \rightarrow V' \;\&\; lgth(V) = lgth(V')))$.

Then it suffices to construct $(\mu_i, e_i)_{i\in\omega}$ according to the following rule :

from μ_i, chose a transition $\mu_i \xrightarrow{e_i} \mu_{i+1}$, if any, with e_i equal to $\{ \nu_R, \overline{\nu}_S \}$ for R some minimal element in the set $\{ R \in ID \mid (\exists \mu)(\mu_R \in Int(\mu_i)) \}$.

It is clear from lemmas 2.3 and 2.6 that the computation thereby obtained must be fair □

Next subsection extends the definitions given so far to the case of open systems.

2.3. OPEN COMPUTATIONS

We introduce open computations as a kind of projections of closed parallel computations.

definition 2.3 Given a closed parallel computation $\mathcal{C} = ((p_i \mid q_i), e_i)_{i\in\omega}$, let $\mathcal{C}_\ell$ and $\mathcal{C}_r$ be the respective sequences $(p_i, (e_i/\leftarrow))_{i\in\omega}$, $(q_i, (e_i/\rightarrow))_{i\in\omega}$ where e/S equals $\{\mu_R \mid \mu_{SR} \in e\}$. For $\mathcal{S}$ in $\{\mathcal{C}_\ell, \mathcal{C}_r\}$, say $\mathcal{S} = (r_i, e'_i)_{i\in\omega}$, set down $|\mathcal{S}| = \sqcup\{i+1 \mid r_i \xrightarrow{e'_i} r_{i+1}\}$ and define $\mathrm{comp}(\mathcal{S})$ as the sequence obtained from $\mathcal{S}$ by erasing pairs (r_i, e'_i) for every $i < |\mathcal{S}|$ such that $e'_i = \emptyset$. Now, $\mathrm{comp}(\mathcal{C}_\ell)$ and $\mathrm{comp}(\mathcal{C}_r)$ are the *left* resp. *right projections* $\mathcal{C}_\leftarrow$ and $\mathcal{C}_\rightarrow$ of $\mathcal{C}$.

definition 2.4 For p_0 in T_Σ, the associated set FAIR(p_0) of *open fair computations* is the set of left projections $\mathcal{C}_\leftarrow$ of the closed computations $\mathcal{C}$ in the set $\bigcup \mathrm{fair}(p_0 \mid q_0)$, $q_0 \in T_\Sigma$. For any such closed computation $\mathcal{C}$, the right projection $\mathcal{C}_\rightarrow$ is called a *co-computation* of p_0.

If $\mathcal{C} \in \mathrm{fair}(p_0 \mid q_0)$, then for obvious reasons of symmetry, $\mathcal{C}_\rightarrow$ and $\mathcal{C}_\leftarrow$ are also an open fair resp. a co-computation of q_0. A few simple properties of

ofer fair computations are stated below after complementary notations are given. The proofs of the propositions are deferred and will appear in subsection 2.4.

Conventions

- name is the subscript erasing function which sends actions to action names (e.g. $\bar{\lambda} = \text{name}(\bar{\lambda}_R)$), and names is the natural set extension of name.
- $E = E_0 \cup E_1 \cup E_2$ where $E_0 = \{\emptyset\}$.
- η is the homomorphism which sends sequences $(e_i)_{i\in\omega}$ of actions / communications from E^ω to $\Lambda^\infty = \Lambda^* \cup \Lambda^\omega$ according to

  $\eta(e_i) = \epsilon$ if $e_i \in E_0$,

  $\eta(e_i) = \lambda$ if $e_i = \{\lambda_R\} \in E_1$,

  $\eta(e_i) = \epsilon$ if $e_i = \{\mu_R, \bar{\mu}_S\} \in E_2$.
- For every word $\rho \in \Lambda^\infty$, Act(ρ) resp. Ult(ρ) denote the subsets of names $\lambda \in \Lambda$ which occur, resp. infinitely often occur in ρ.

proposition 2.1. given p_0 in T_Σ, FAIR(p_0) is equal to the set of sequences $(p_i, e'_i)_{i\in\omega}$, with $p_i \in T_\Sigma$ and $e'_i \in E$, for which the following conditions 1 to 4 are satisfied :

1- $(\forall i)((p_i \xrightarrow{e'_i} p_{i+1}) \vee (e'_i = \emptyset \;\&\; p_i = p_{i+1}))$

2- $(e'_i)_{i\in\omega} \in (E_1 \cup E_2)^\omega \cup (E_1 \cup E_2)^* E_0^\omega$

3 - $(\forall R \in iD)(\exists j)(\forall k \geq j)(Int_R(p_k) = \emptyset)$

4 - $(\exists R)(\exists j)(\forall k \geq j)(\lambda \in Ext_R(p_k)) \Rightarrow$

$\lambda \notin Ult(\eta((e'_i)_{i \in \omega}))$ □

(Henceforth, we call an <u>open computation</u> every sequence $(p_i, e'_i)_{i \in \omega}$ which satisfies 1 and 2).

<u>corollary 1</u> For any program $p_0 \in T_\Sigma$, the set $fair(p_0)$ of closed fair computations of p_0 is the subset of $FAIR(p_0)$ made out of the computations $(p_i, e_i)_{i \in \omega}$ for which every e_i is either a pair of actions in E_2 or is the empty set of actions.

<u>corollary 2</u> Every finite prefix of an open computation, say $(p_i, e_i)_{i < m}$, may be extended into an open fair computation by chosing $(p_i, e_i)_{i \geq m}$ in the set $fair(p_m)$ - which is not empty by lemma 2.8 -

Let us conduct a little further the investigation of the relations between open and closed fair computations. Given programs p_0 and q_0, an important issue is to determine under which conditions a pair of open computations $\mathcal{E}'$, $\mathcal{E}''$ of p_0 resp. q_0 are the respective projections of some closed fair computation of $p_0 | q_0$. Since definition 2.3 applies as well to open parallel computations, the question may also be asked for

open computations of $p_0 \mid q_0$. By the first corollary above, an answer to the latter problem will also provide an answer to the former. The corresponding statements follow.

proposition 2.2 Let $\mathcal{C} \in FAIR(p \mid q)$ be an open fair computation, then $\mathcal{C}_{\leftarrow} \in FAIR(p)$ and $\mathcal{C}_{\rightarrow} \in FAIR(q)$.

proposition 2.3 Let the pair of open fair computations $\mathcal{E}' = (p_i, e'_i)_{i \in \omega}$ and $\mathcal{E}'' = (q_i, e''_i)_{i \in \omega}$. There exists $\mathcal{C}$ in $FAIR(p_0 \mid q_0)$ with projections $\mathcal{C}_{\leftarrow} = \mathcal{E}'$ and $\mathcal{C}_{\rightarrow} = \mathcal{E}''$ iff. implications 1 and 2 below are satisfied for every $\lambda \in \Lambda$ and $R \in ID$:

1 - $\lambda \in \bigsqcup \{ \bigcap_{k \geq i} Ext_R(p_k) \mid i \in \omega \} \Rightarrow a \,\&\, b$

a) $\lambda \notin ult(\eta((e''_i)_{i \in \omega}))$

b) $\bar{\lambda} \notin \sqcap \{ \bigcup_{k \geq i} names(Ext(q_k)) \mid i \in \omega \}$

2 - $\lambda \in \bigsqcup \{ \bigcap_{k \geq i} Ext_R(q_k) \mid i \in \omega \} \Rightarrow a' \,\&\, b'$

a') $\lambda \notin ult(\eta((e'_i)_{i \in \omega}))$

b') $\bar{\lambda} \notin \sqcap \{ \bigcup_{k \geq i} names(Ext(p_k)) \mid i \in \omega \}$ □

In fact, the proof of proposition 2.3 allows to state a more precise result, using the following ...

definition 2.5 Let the open computations $\mathcal{E}' = (p_i, e'_i)_{i \in \omega}$ and $\mathcal{E}'' = (q_i, e''_i)_{i \in \omega}$, of respective sizes $|\mathcal{E}'|$ and $|\mathcal{E}''|$ as in definition 2.3 . A pair (f, g) of monotonous onto functions

from ω to ω is a <u>scheduling</u> of (Σ', Σ'') iff $\forall i$:

0 - $f(i) \neq f(i+1) \;\vee\; g(i) \neq g(i+1)$

1 - $f(i) \geq |\Sigma'| \Rightarrow f(i+1) = f(i)+1$

2 - $g(i) \geq |\Sigma''| \Rightarrow g(i+1) = g(i)+1$

3 - $(f(i) \neq f(i+1) \leq |\Sigma'|) \;\&\; (g(i) \neq g(i+1) \leq |\Sigma''|)$
$\Rightarrow ((e'_{f(i)} = \{\lambda_R\}) \;\&\; (e''_{g(i)} = \{\bar{\lambda}_S\})$, some λ, R, S) .

Then, the $_f{\times}_g$ <u>product</u> of Σ' and Σ'' is the open computation $\Sigma' {}_f{\times}_g \Sigma'' = ((p_{f(i)} \mid q_{g(i)}), e_i)_{i \in \omega}$ where

e_i = if $f(i) \neq f(i+1) \;\&\; g(i) \neq g(i+1)$
then $\leftarrow(e'_{f(i)}) \cup \rightarrow(e''_{g(i)})$
or else if $f(i) \neq f(i+1)$ then $\leftarrow(e'_{f(i)})$
or else if $g(i) \neq g(i+1)$ then $\rightarrow(e''_{g(i)})$.

<u>proposition 2.4.</u> Under the conditions of proposition 2.3, each scheduling (f,g) of (Σ', Σ'') determines an open fair computation $\mathcal{C} = \Sigma' {}_f{\times}_g \Sigma''$ with projections $\mathcal{C}_\leftarrow = \Sigma'$ and $\mathcal{C}_\rightarrow = \Sigma''$. Conversely, every open fair computation $\mathcal{C}$ with projections $\mathcal{C}_\leftarrow = \Sigma'$ and $\mathcal{C}_\rightarrow = \Sigma''$ may be written $\Sigma' {}_f{\times}_g \Sigma''$ for some scheduling (f,g) of (Σ', Σ'') . □

<u>corollary 3</u> Given a pair of open fair computations Σ', Σ'' as above, Σ' and Σ'' are the respective projections of some closed fair computation iff they satisfy the conditions of proposition 2.3 together with the requirement 3 - $\eta((e'_i)_{i\in\omega}) = \overline{\eta((e''_i)_{i\in\omega})}$.

To sum up our current results, we have shown that closed fair computations may be generalized into open fair computations amenable to the principle of composition. Moreover, we have gained some insight into the possible objects of fully abstract models, since proposition 2.3 tells us which properties of a computation may influence the associated environment. From the symmetry between open computations and co-computations, this remark holds both for operational models, which abstract open fair computations, and for observational models, which abstract co-computations.

2.4 PROOFS OF PROPOSITIONS

proof of proposition 2.1

We shall content ourselves with showing that the set of the open computations of p_0 which satisfy the conditions of the proposition is included in $FAIR(p_0)$: by lemmas 2.4 to 2.6, the converse inclusion is straightforward. So, suppose that conditions 3 and 4 are satisfied for some $\mathcal{E}'$ equal to $(p_i, e'_i)_{i \in \omega}$. Set down $p = \eta((e'_i)_{i \in \omega})$ and $Ult(p) = \{\lambda_1, \ldots, \lambda_n\}$. Also define $m = \min\{j \geq 0 \mid (p>j) \in (Ult(p))^\infty\}$, where $(p>j)$ is the quotient of p by its left

factor with length j. Now consider the following series of programs $(q_i)_{i \leq m}$, where $\rho(j)$ is the $j^{\underline{th}}$ occurence in ρ :

$q_{m-k} = \langle \overline{\rho(m-k+1)} \rangle (\dots (\langle \overline{\rho(m)} \rangle (q_m)) \dots)$,

$q_m = NIL$ if $|\rho| < \omega$ or else

$q_m = (\tau \mid s)\,[\pi]$ where :

τ is the program $\langle \mathcal{C} \Rightarrow \bar{\alpha}_1 : \mathcal{C}, \dots, \bar{\alpha}_n : \mathcal{C} \rangle$,

s is the program $\langle \bar{\mathcal{C}} \Rightarrow \bar{\mathcal{C}} : \bar{\mathcal{C}} \rangle$,

π is a renaming function which sends α_i to λ_i for $1 \leq i \leq n$ and is undefined for $\mathcal{C}$.

For $k < m$, define $e''_k = \{ \overline{\rho(k+1)}_{\epsilon} \}$.

Clearly, there exist a unique sequence $(q_i)_{i \geq m}$ of programs and a unique sequence $(e''_i)_{i \geq m}$ in E^ω such that $(q_i, e''_i)_{i \in \omega}$ is an open computation $\mathcal{E}''$ which satisfies

$e''_{m+9i+3j-2} = \{ \overline{\rho(m+3i+j)}_{\leftarrow(\rightarrow)\, 3i+j} \}$

if ρ is an infinite sequence.

That computation is obviously fair : one has indeed $\bigcap \{ Int_R(q_k) \mid k \geq j \} = \emptyset$ for every R and j.

Also clearly, there exists some scheduling (f, g) of $(\mathcal{E}', \mathcal{E}'')$ such that the ${}_f{\times}_g$ product of $\mathcal{E}', \mathcal{E}''$ is a closed computation, say $\mathcal{E}$. We claim that $\mathcal{E}$ is fair.

To prove the claim, suppose that condition 3 of definition 2.2 is not satisfied for $\mathcal{E}$. Then one among the following cases must be found for some $S \in ID$:

- $(\forall j)(\exists k \geq j)(Int_S(q_{g(k)}) \neq \emptyset)$:
  impossible from the construction of the q_i
- $(\forall j)(\exists k \geq j)(Int_S(r_{f(k)}) \neq \emptyset)$:
  impossible by condition 3 in the proposition
- $(\forall j)(\exists k \geq j)(Ext_S(r_{f(k)}) \cap \{\lambda_1 \ldots \lambda_n\} \neq \emptyset)$:
  then there exists some λ in $Ult(\rho)$ such that
  $(\forall j)(\exists k \geq j)(\lambda \in Ext_S(r_{f(k)}))$, which is impossible
  by condition 4 in the proposition.

proof of proposition 2.2 : straightforward from proposition 2.1 and lemmas 2.4 to 2.6 by the following remark : for $R \neq S$, $\lambda \in Ext_R(r)$ and $(p|q) \xrightarrow{\lambda \leftarrow S\ ,\ \bar{\lambda} \rightarrow T} (p'|q')$ together imply $\lambda \in Int_{\leftarrow R}(p|q)$.

proof of proposition 2.3.

only if part Assume $\mathcal{E}' = \mathcal{C}_{\leftarrow}$ and $\mathcal{E}'' = \mathcal{C}_{\rightarrow}$ for some open computation $\mathcal{C}$ in $FAIR(p_0|q_0)$, let $\mathcal{C} = ((p'_i|q'_i), e_i)_{i \in \omega}$. Clearly, there exists some scheduling (f,g) of $(\mathcal{E}', \mathcal{E}'')$ such that $\mathcal{C} = \mathcal{E}'_f \times_g \mathcal{E}''$. It is enough to prove that condition 1 in the statement of the proposition is satisfied for every R, since a similar argument may then be given to establish condition 2.

So, consider some process identifier $R \in ID$, and assume $\lambda \in \bigsqcup \{ \bigcap_{k \geq i} Ext_R(p_k) \mid i \in \omega \}$.

Then 3 holds :

3 - $\lambda_{\leftarrow R} \in \bigsqcup \{ \bigcap_{k \geq i} Ext(p_{f(k)} \mid q_{g(k)}) \mid i \in \omega \}$,

and since $\mathcal{C}$ is an open fair computation, proposition 2.1 now shows :

4 - $(\exists l)(\forall k \geq l)(Int_{\leftarrow R}(p_{f(k)} \mid q_{g(k)}) = \emptyset)$

5 - $\lambda \notin Ult(q((e_i)_{i \in \omega}))$.

From 3, 4 and by lemma 2.4 :

$(\exists l)(\forall k \geq l)(\forall S)(\bar{\lambda}_S \notin Ext(q_{g(k)}))$,

whence b is satisfied.

There remains to establish a.

From 5, one can find l such that

$(\forall k \geq l)(\forall S)(e_k \neq \{\lambda_S\})$.

Hence, either a is satisfied, or

$(\forall i)(\exists k \geq i)(\exists T)(e'_k = \{\bar{\lambda}_T\})$,

for f and g are monotonous onto functions.

Now, by proposition 2.1, the second case cannot occur since $\mathcal{E}' \in FAIR(p_0)$, and from the assumption on λ.

<u>if part</u> Assume that conditions 1 and 2 are satisfied for every $R \in ID$. Consider $\mathcal{C}$ equal to $\mathcal{E}'_f \times_g \mathcal{E}''$ for some scheduling (f, g) of $(\mathcal{E}', \mathcal{E}'')$, then clearly $\mathcal{C}_{\leftarrow} = \mathcal{E}'$ and $\mathcal{C}_{\rightarrow} = \mathcal{E}''$. We claim that $\mathcal{C}$ is a fair computation. To prove the claim, suppose for a moment

$\mathcal{C} \notin FAIR(p_0 \mid q_0)$ and seek for a contradiction.

Set down $\mathcal{C} = ((p_{f(i)} \mid q_{g(i)}), e_i)_{i \in \omega}$.

By proposition 2.1, one among the following assertions 3 or 4 must be true for some $\lambda \in \Lambda$ and $R \in iD$:

3 - $(\forall j)(\exists k \geq j)(Int_R(p_{f(k)} \mid q_{g(k)}) \neq \emptyset)$

4 - $(\exists j)(\forall k \geq j)(\lambda \in Ext_R(p_{f(k)} \mid q_{g(k)}))$
& $(\forall i)(\exists k \geq i)(\exists S \in iD)(e_k = \{\lambda_S\})$.

For the reason of symmetry, we are free to assume $R = \leftarrow T$ for some $T \in iD$.

Suppose 3. From $\mathcal{E}' \in FAIR(p_\circ)$ and by proposition 2.1, the following assertion must be true :

$(\forall j)(\exists k \geq j)(\exists \lambda)(\exists S \in iD)$:

$\lambda_T \in Ext(p_{f(k)})$ & $\bar{\lambda}_S \in Ext(q_{g(k)})$.

Hence, by lemma 2.7 : $\lambda \in \bigsqcup \{ \bigcap_{k \geq i} Ext_T(p_k) \mid i \in \omega \}$ and $\bar{\lambda} \in \bigsqcap \{ \bigcup_{k \geq i} names(Ext(q_k)) \mid i \in \omega \}$,
and a contradiction of condition 1 has been reached.

Suppose 4, then by 1.a :

$(\forall j)(\exists k \geq j)(\forall S)(e''_{g(k)} \neq \{\lambda_S\})$.

Since f is a monotonous onto function, the following assertion must be true :

$(\forall j)(\exists k \geq j)(\exists S)(e'_k = \{\lambda_S\})$.

From $\mathcal{E}' \in FAIR(p_\circ)$ and by proposition 2.1, it follows that condition 1 is again contradicted.

3. AN OPERATIONAL MODEL

The purpose of the section is to provide T_Σ with a language theoretic model, whose objects are straightforward abstractions of the operational behaviours of programs, i.e. of their open fair computations.

3.1. COMPUTATION HISTORIES

definition 3.1. Let $\mathcal{E} = (p_i, e_i)_{i \in \omega}$ be an open fair computation, the history abs $(\mathcal{E})$ of $\mathcal{E}$ is the triple $\langle \delta, \rho, d \rangle$ with elements :

$$\delta = \text{names}\left(\sqcup \{ \bigcap_{j \geq k} \text{Ext}(p_j) \mid k \in \omega \} \right)$$

$$\rho = \eta((e_i)_{i \in \omega})$$

$$d = \sqcap \{ \bigcup_{j \geq k} \text{names}(\text{Ext}(p_j)) \mid k \in \omega \} \setminus \delta .$$

Some comments follow here. Given $\mathcal{E}$ in FAIR(p_0), abs$(\mathcal{E})$ captures those features of the computation which may have influence on the unknown environment q_0 in the course of some closed fair computation of $p_0 \mid q_0$. Let abs$(\mathcal{E})$ be equal to $\langle \delta, \rho, d \rangle$, then ρ reflects the sequence of the effective interactions between the program p_0 and the environment, δ reflects the unsuccessful interaction attempts of the program, and d identifies the interactions infinitely

often offered to the environment . We insist on the fact that the withdrawn details of the open computation Σ exert no influence on the environment under our general assumption of fair asynchrony , which fact is made clear by the first corollary of proposition 2.3 . Our intend is to construct a model of T_Σ in which a program p is represented by a set $Abs(p)$ of computation histories according to the following ...

<u>definition 3.2</u> $Abs(p) = \{ abs(\Sigma) \mid \Sigma \in FAIR(p) \}$.

These set interpretations may be viewed as languages : by lemmas 2.1 and 2.2 , $Abs(p)$ can always be written as a finite sum $\sum_i \langle \delta_i , \mathcal{L}_i , d_i \rangle$ where the $\mathcal{L}_i$ are infinitary languages defined on a finite subset of Λ . These sets might also be seen as the morphic images of other languages , defined on an alphabet whose letters , called history items, are triples $\langle \delta, p, d \rangle$ such that $p \in \Lambda \cup \{\epsilon\}$.

Now, set theoretic and in particular language theoretic models can be constructed without caring about the continuity of operators , for it is well known that least and greatest fixpoints of monotonous operators always exist in complete lattices. Before we take advantage of this fact , the technical framework is made more explicit in the next subsection .

3.2. THE DOMAIN OF THE MODEL

In all the sequel, L denotes a finite subset of Λ such that $L = sym(L)$, and Sorts is the set made out of the so-called sorts L. The domain of the forthcoming model is the powerset $\mathcal{P}(\mathcal{H})$ of $\mathcal{H}$, the set of histories given by the following...

definition 3.3. $\mathcal{H}_L$, the set of L-sorted histories, is the collection of the triples $\langle \delta, \rho, d \rangle$ which satisfy : $\delta \subseteq L$ & $\rho \in L^\infty$ & $(Ult(\rho) \subseteq d \subseteq L)$ & $(d \cap (\delta \cup \bar{\delta}) = \emptyset)$. $\mathcal{H}$, the set of histories, is the union $\bigcup \mathcal{H}_L$, $L \in$ Sorts. $\langle \delta, \mathcal{L}, d \rangle$ and $\{ \langle \delta, \rho, d \rangle \mid \rho \in \mathcal{L} \}$ are equivalent notations for subsets of histories.

Histories are provided with prefixing, renaming and composition operators as follows.

definition 3.4 To each action name $\lambda \in \Lambda \cup \{\epsilon\}$, we associate the operator $\lambda :$ from $\mathcal{H}$ to $\mathcal{P}(\mathcal{H})$ such that $\lambda : \langle \delta, \rho, d \rangle$ is the singleton set $\langle \delta, \lambda\rho, d \rangle$.

definition 3.5 To each function $\pi \in Ren$, we associate the function $[\![\pi]\!]$ from $\mathcal{H}$ to $\mathcal{P}(\mathcal{H})$ such that : $[\![\pi]\!](\langle \delta, \rho, d \rangle)$ is the empty subset of $\mathcal{H}$ if $(\rho = \rho' \lambda \rho'' \;\&\; \pi(\lambda) = \tau)$ for some $\lambda \in \Lambda$, or else is the singleton set $\langle \pi\delta, \pi\rho, \pi d \rangle$.

definition 3.6 given histories h' and h'' in $\mathcal{H}$, let $h' = \langle \delta', \rho', d' \rangle$ and $h'' = \langle \delta'', \rho'', d'' \rangle$, these histories are *compatible* ($h' \# h''$) iff

1 - $\delta' \cap Ult(\rho'') = \emptyset = \delta'' \cap Ult(\rho')$

2 - $\delta' \cap (\bar{d}'' \cup \bar{\delta}'') = \emptyset = \delta'' \cap (\bar{d}' \cup \bar{\delta}')$

The *parallel composition* $h' \| h''$ of h', h'' is either the empty subset of $\mathcal{H}$ if $\neg(h' \# h'')$ or else is the set of histories $\langle \delta' \cup \delta'', \rho' \| \rho'', (d' \setminus \delta'') \cup (d'' \setminus \delta') \rangle$ for $\|$ the operator on words given below.

definition 3.7 Let words ρ', ρ'' in L^∞, their parallel composition $\rho' \| \rho''$ is the set of the words $\rho = \rho_1 \rho_2 \cdots \rho_i \cdots$ for which there exist corresponding factorizations $\rho'_1 \rho'_2 \cdots \rho'_i \cdots$ resp. $\rho''_1 \rho''_2 \cdots \rho''_i \cdots$ of ρ' resp. ρ'' on $(L \cup \{\epsilon\})$ such that for every i :

- $(\rho'_i \neq \epsilon \ \& \ \rho''_i \neq \epsilon) \Rightarrow (\rho'_i = \bar{\rho}''_i \ \& \ \rho_i = \epsilon)$
- $(\rho'_i = \epsilon \vee \rho''_i = \epsilon) \Rightarrow \rho_i = \rho'_i \rho''_i$.

Set extensions of the operators $\lambda :$, $\|$ and $\mathbb{I}\pi\mathbb{I}$ will be freely used down here with $\mathcal{P}(\mathcal{H})$ as their co-domain. For instance, $\mathbb{I}\pi\mathbb{I}(\{h_1, h_2\})$ amounts implicitly to $\mathbb{I}\pi\mathbb{I}(h_1) \cup \mathbb{I}\pi\mathbb{I}(h_2)$.

3.3 . THE MODEL

To each program in T_Σ, say p, we associate an operational meaning $\mathcal{M}(p)$ in the form of a

set of histories. More precisely, if L is the sort of p, then $\mathcal{M}(p)$ is a subset of $\mathcal{HC}_L$.

<u>definition 3.8</u>. The <u>operational meaning</u> function $\mathcal{M} : T_\Sigma \longrightarrow \mathcal{P}(\mathcal{HC})$ is inductively defined by the following equations, where we let $u_i = \lambda\lambda_i\lambda'_i$, $\mathcal{L} = (\sum_{i=1}^{n} u_i)^*$, $\mathcal{L}_I = \bigcap_{i\in I} (\mathcal{L} u_i)^\omega$ for $I \subseteq \{1 \dots n\}$, and use the additive notation for set unions :

$\mathcal{M}(\mathrm{NIL}) = \langle \emptyset, \epsilon, \emptyset \rangle$,

$\mathcal{M}(\langle \lambda_1, \dots, \lambda_n \rangle (t_1, \dots, t_n)) =$

$\quad \langle \{\lambda_1 \dots \lambda_n\}, \epsilon, \emptyset \rangle + \sum_{i=1}^{n} (\lambda_i : \mathcal{M}(t_i))$,

$\mathcal{M}(t_1 \mid t_2) = \mathcal{M}(t_1) \parallel \mathcal{M}(t_2)$,

$\mathcal{M}(t[\pi]) = [\![\pi]\!](\mathcal{M}(t))$,

$\mathcal{M}(t[\lambda \triangleright \pi]) = Y(F)$, the greatest fixpoint of

$\quad F(X) = \langle \lambda, \epsilon, \emptyset \rangle + (\lambda : [\![\pi]\!](\mathcal{M}(t) \parallel X))$,

$\mathcal{M}(\langle \lambda \Rightarrow \lambda_1 : \lambda'_1, \dots, \lambda_n : \lambda'_n \rangle) =$

$\quad \langle \{\lambda\}, \mathcal{L}, \emptyset \rangle + \langle \{\lambda_1 .. \lambda_n\}, \mathcal{L}\lambda, \emptyset \rangle + \sum_{i=1}^{n} \langle \{\lambda'_i\}, \mathcal{L}\lambda\lambda_i, \emptyset \rangle$

$+ \sum_{I \neq \emptyset} \langle \emptyset, \mathcal{L}_I, \{\lambda\} \cup \{\lambda_1 .. \lambda_n\} \cup \{\lambda'_i \mid i \in I\} \rangle$.

It should be clear that for any $X \subseteq \mathcal{HC}$, $F(X)$ is a subset of $\mathcal{HC}_L$ for L equal to $\{\lambda, \overline{\lambda}\} \cup \pi(\Lambda)$, a finite subset of Λ by the definition of Ren. However, this fact does not imply the inclusion $\mathcal{M}(t) \subseteq \mathcal{HC}_L$ for L equal to the sort of t, which stronger fact will be proven later on. Also notice that function F is monotone for the inclusion of subsets, whence the

greatest fixpoint $Y(F)$ of F exists and may be written $F^{\alpha}(\mathcal{H})$ for α some sufficiently high ordinal.

The remainder of section 3 intends to show that $\mathcal{M}$ is a terminal model of the <u>operational equivalence</u> $\sim_{Abs}$ on T_{Σ} : $t \sim_{Abs} t'$ iff $Abs(t) = Abs(t')$. This property is also known as <u>full abstraction</u> under the formulation :

$\mathcal{M}(t) = \mathcal{M}(t')$ iff $(\forall \mathcal{C})(Abs(\mathcal{C}(t)) = Abs(\mathcal{C}(t')))$

where $\mathcal{C}$ ranges over the set of Σ - contexts.

By the obvious implication

$\mathcal{M}(t) = \mathcal{M}(t') \Rightarrow \mathcal{M}(\mathcal{C}(t)) = \mathcal{M}(\mathcal{C}(t'))$,

the full abstractness of the model $\mathcal{M}$ w.r.t. $\sim_{Abs}$ will follow as an immediate corollary from the property <u>$\mathcal{M}(t) = Abs(t)$</u>. Before we proceed to establish this identity, next division introduces a new series of definitions which link computations to objects of the model and vice-versa.

3.4. THE FULL APPARATUS OF THE MODEL

<u>definition 3.9</u> $\mathcal{J}_L$, the set of <u>L-sorted history items</u>, is the collection of the triples $\langle \delta, \rho, d \rangle$ which satisfy $\delta \subseteq L \;\&\; \rho \in (L \cup \{\varepsilon\}) \;\&\; (Act(\rho) \subseteq d \subseteq L)$.

<u>hint</u> : each item represents an elementary step of computation, ρ is an action (a product of actions),

d displays (the union of) the corresponding alternative(s), δ displays the actions which freeze at the considered instant in time.

<u>definition 3.10</u>. We let <u>trace</u> denote the function from $FAIR(T_\Sigma)$ to $\bigcup_L (\mathcal{I}_L^\omega)$ which sends each open fair computation, say $\xi = (p_i, e_i)_{i \in \omega}$, to the infinite word $trace(\xi)$ given by $((h_i)_{i \in \omega})$ with h_i equal to $\langle \delta_i, p_i, d_i \rangle$ as follows:

- $p_i = \eta(e_i)$
- $d_i = names\,(\bigcup_R Ext_R(p_i) \mid R \underline{acts\ at}\ i)$

where R <u>acts at</u> i iff $\mu_R \in e_i$ for some μ

- $\delta_i = names\,(\bigcup_R Ext_R(p_i) \mid R \underline{freezes\ at}\ i)$

where R <u>freezes at</u> i iff

$i = min\,\{j \mid Ext_R(p_j) \neq \emptyset\ \&\ (R \underline{acts\ at}\ k \Rightarrow k < j)\}$

(Lemmas 2.1 and 2.2 show that trace is a well defined function).

<u>definition 3.11</u>. $\varphi : (\bigcup_L \mathcal{I}_L^\omega) \longrightarrow (\bigcup_L \mathcal{H}_L)$ is the sort preserving function which sends each infinite sequence of history items, let $A = \langle \delta_i, p_i, d_i \rangle_{i \in \omega}$, to the history $\varphi(A)$ given by the following statements, where $\sigma \notin \mathcal{H}$ denotes the undefined history:

$\varphi(A) = \sigma$ if $\delta_i \cap \bar{\delta}_j \neq \emptyset$ for some $i \neq j$,

or else letting

- $\delta = \bigsqcup \{\delta_i \mid i \in \omega\}$

- $\rho = \rho_1 \rho_2 \cdots \rho_i \cdots$
- $d = \sqcap \{ \bigcup_{i \geq j} d_i \mid j \in \omega \}$

$\varphi(A) = \sigma$ *if* $(d \cap \bar{\delta} \neq \emptyset \vee Ult(\rho) \cap \delta \neq \emptyset)$, *or else*

$\varphi(A) = \langle \delta, \rho, d \setminus \delta \rangle$

hint: *the undefined history is reached whenever complementary actions* λ *and* $\bar{\lambda}$ *freeze at different instants in time or some action capability* λ *is left intact in spite of an infinite number of concurrent operations* λ *or escapes from* $\bar{\lambda}$.

proposition 3.1. *For every* $\mathcal{E}$ *in* $FAIR(T_\Sigma)$, $\varphi(trace(\mathcal{E})) = abs(\mathcal{E})$.

proof. *Set down* $abs(\mathcal{E}) = \langle \delta, \rho, d \rangle$, *and let* $trace(\mathcal{E}) = (h_i)_{i \in \omega}$ *with the* h_i *as in definition 3.10, then* $\rho = \rho_1 \rho_2 \cdots \rho_i \cdots$ *obviously, and* $\delta = \sqcup \{ \delta_i \mid i \in \omega \}$ *by lemmas 2.5 and 2.6*. *By these lemmas again, let* $\lambda \in names(Ext(\rho_j))$ *for some* j, *then there exists* $R \in ID$ *such that either a or b is satisfied*:

a - R *freezes at* i *and* $\lambda \in Ext_R(\rho_i)$ *for some* $i \leq j$,

b - R *acts at* i *and* $\lambda \in Ext_R(\rho_i)$ *for some* $i \geq j$.

Therefore, $d = (\sqcap \{ \bigcup_{i \geq j} d_i \mid j \in \omega \}) \setminus \delta$.

There remains to prove that $\varphi(trace(\mathcal{E}))$ *is defined, whence* $\varphi(trace(\mathcal{E})) = \langle \delta, \rho, d \rangle \in \mathcal{HG}$.

Suppose $\delta_i \cap \bar{\delta}_j \neq \emptyset$ *for* $i \neq j$, *or*

$\bar{\delta}_j \cap (\Pi \{ \bigcup_{i \geq k} d_i \mid k \in \omega \}) \neq \emptyset$:
by lemma 2.4 and condition 3 in proposition 2.1,
both situations are impossible.
Condition 4 in the same proposition indicates also
that $(Ult(\rho) \cap \delta)$ must be empty $\square$

corollary 1. Let $\rho \in T_\Sigma$, if L is the minimal sort
such that $Abs(\rho) \subseteq \mathcal{H}_L$, then L is also the minimal
sort such that $\rho \in L^\infty$ for every $\langle \delta, \rho, d \rangle$ in $Abs(\rho)$.
– from the definitions of trace and Ext_R, and by
the second corollary of proposition 2.1 –

corollary 2. Let $\rho \in T_\Sigma$, if $L = sort(\rho)$ then
$Abs(\rho)$ is a subset of $\mathcal{H}_L$.
– by lemma 2.2 –

proposition 3.2 $\mathcal{H} = \varphi(trace(FAIR(T_\Sigma)))$.
proof. Given some history $\langle \delta, \bar{\rho}, d \rangle$ in $\mathcal{H}$, set down
$Ult(\bar{\rho}) = \{ \bar{\lambda}_1, \dots, \bar{\lambda}_n \}$, $d \setminus Ult(\bar{\rho}) = \{ \mu_1 \dots \mu_m \}$, and
$\delta = \{ \nu_1, \dots, \nu_\ell \}$. Let program q_0 and let computation
$\mathcal{E}'' = (q_i, e''_i)_{i \in \omega}$ be replicated from the proof 2.1.
Define $\rho_0 = \langle \nu_1, \dots, \nu_\ell \rangle (NIL, \dots, NIL)$, and
$\tau_0 = (\langle \alpha \Rightarrow \alpha : \alpha, \mu'_1 : \gamma, \dots, \mu'_m : \gamma \rangle \mid \langle \bar{\alpha} \Rightarrow \bar{\alpha} : \bar{\alpha} \rangle) [\pi']$
where $\pi'(\alpha) = \tau$ and $\pi'(\mu'_i) = \mu_i$ for i in $\{1 \dots m\}$.
Now define $\mathcal{E}'_1$ as the 0-sized computation from ρ_0,
and $\mathcal{E}'_2$ as the unique closed ω-sized computation

from r_0. Owing to the assumption $\langle \delta, \bar{\rho}, d \rangle \in \mathcal{H}$ and by proposition 2.3, there exists some computation $\mathcal{E}$ in FAIR $((p_0 \mid r_0) \mid q_0)$ with its left.left, left.right and right projections respectively equal to $\mathcal{E}'_1$, $\mathcal{E}'_2$ and $\mathcal{E}''$. Clearly, by the construction of $\mathcal{E}$, $\varphi(\text{trace}(\mathcal{E})) = \langle \delta, \bar{\rho}, d \rangle$. □

The remaining statements show that φ is indeed a sort preserving surjective <u>morphism</u> from $\bigcup_L \mathcal{I}_L^\omega$ to $\bigcup_L \mathcal{H}_L$, once $\mathcal{I}_L^\omega$ has been equipped with prefixing, renaming and composition operators. Natural candidates for the first and second families of operations are as follows.

<u>definition 3.12</u>. To each $a \in \mathcal{I}_L$, we associate the function from $\mathcal{I}_L^\omega$ to $\mathcal{P}(\mathcal{I}_L^\omega)$ such that $a:A$ is the empty set if $\varphi(aA) = \sigma$, or else is the singleton set $\{aA\}$.

<u>definition 3.13</u>. To each $\pi \in \text{Ren}$, we associate the function from $\mathcal{I}_L^\omega$ to $\bigcup_L \mathcal{P}(\mathcal{I}_L^\omega)$ such that, letting $A = \langle \delta_i, \rho_i, d_i \rangle_{i \in \omega}$, $[\![\pi]\!] A$ is the empty set if $\varphi(A) = \sigma$ or $\pi(\rho_i) = \tau$ for some i, or else is the singleton set $\{ \langle \pi\delta_i, \pi\rho_i, \pi d_i \rangle_{i \in \omega} \}$.

As regards the composition operator, we need two auxiliary definitions.

definition 3.14. Let the history items $a' = \langle \delta', \rho', d' \rangle$ and $a'' = \langle \delta'', \rho'', d'' \rangle$ in $\mathcal{J}_L$, their *synchronous product* $a' \times a''$ is defined iff

$(\delta' \cap \bar{\delta}'' = \emptyset)$ & $(\rho' = \epsilon \vee \rho' = \bar{\rho}'' \vee \rho'' = \epsilon)$

and is then equal to $\langle \delta' \cup \delta'', \rho, d' \cup d'' \rangle$ with $\rho = \epsilon$ if $\rho' = \bar{\rho}''$ or else $\rho = \rho' \rho''$. The product operation naturally extends from $\mathcal{J}_L$ to $\mathcal{J}_L^{\omega}$.

definition 3.15. Let $A = \langle \delta_i, \rho_i, d_i \rangle_{i \in \omega}$ in $\mathcal{J}_L^{\omega}$. Given f, a monotonous onto function from ω to ω, $A \square f$ is the infinite word $\langle \delta'_i, \rho'_i, d'_i \rangle_{i \in \omega}$ s.t.

- $\delta'_i =$ if $(i = 0 \vee f(i) \neq f(i-1))$ then $\delta_{f(i)}$ else $\emptyset$,
- $\rho'_i =$ if $(f(i) \neq f(i+1))$ then $\rho_{f(i)}$ else ϵ,
- $d'_i =$ if $(f(i) \neq f(i+1))$ then $d_{f(i)}$ else $\emptyset$.

definition 3.16. Given words A and B in $\mathcal{J}_L^{\omega}$, their *asynchronous composition* $A \| B$ is the set made out of the words $C \in \mathcal{J}_L^{\omega}$ which satisfy $\varphi(C) \neq \sigma$ and $C = (A \square f) \times (B \square g)$ for some pair (f, g) of monotonous onto functions from ω to ω such that $(f(i) \neq f(i+1)) \vee (g(i) \neq g(i+1))$ for every i.

proposition 3.3. For each particular sort L, the set extension of $\varphi : \mathcal{J}_L^{\omega} \rightarrow \mathcal{HC}_L$ may be considered as a morphism in the following sense :

$\varphi(\langle \delta, \rho, d \rangle : A) \in \rho : \varphi(A) \subseteq \varphi(\langle \emptyset, \rho, d \rangle : A)$,

$\varphi([\![\pi]\!] A) = [\![\pi]\!] \varphi(A)$, $\varphi(A \| B) = \varphi(A) \| \varphi(B)$ $\square$

3.5. FULL ABSTRACTNESS OF THE MODEL

Before we establish the identity $\mathcal{M} = Abs$, some immediate consequences are worth noting. For instance, the corollaries of proposition 3.1 remain valid once $\mathcal{M}$ has been substituted for Abs, and $\mathcal{M}(t)$ differs from $\emptyset$ for every $t \in T_\Sigma$ because $Abs(t)$ has been shown non empty by lemma 2.8. Given $<\delta, \rho, d>$ in $\mathcal{M}(t)$, it also appears by the second corollary of proposition 2.1 that some history $<\delta', \rho', d'>$ occurs in $\mathcal{M}(t)$ for each proper prefix ρ' of ρ. The section ends with the proof of the following ...

theorem 3.1. $\mathcal{M} = Abs$

proof : obvious from propositions 3.4 and 3.5 below □

proposition 3.4 $\forall t \in T_\Sigma$, $Abs(t) \subseteq \mathcal{M}(t)$

proof. By proposition 3.1, there suffices to show that $\varphi(\text{trace}(\mathcal{E}))$ belongs to $\mathcal{M}(t)$ for every $\mathcal{E} \in FAIR(t)$. Using implicitly the induction on Σ-terms, we establish this fact by case analysis according to the head symbol of term t.

$t = NIL$: immediate.

$t = <\lambda_1, \ldots, \lambda_n>(t_1, \ldots, t_n)$. Now, $FAIR(t)$ is the set $\{(t, \emptyset)^\omega\} \cup \bigcup_i \{(t, \{\lambda_i\}).\mathcal{E}_i \mid \mathcal{E}_i \in FAIR(t_i)\}$.

$\text{trace}((t, \emptyset)^\omega) = <\{\lambda_1 \ldots \lambda_n\}, \epsilon, \emptyset> . <\emptyset, \epsilon, \emptyset>^\omega$,

$\mathrm{trace}((t, \{\lambda_i\}) \cdot \Sigma_i) = \langle \emptyset, \lambda_i, \{\lambda_1 \dots \lambda_n\} \rangle \cdot \mathrm{trace}(\Sigma_i)$.

The desired result follows by proposition 3.3 and from the induction hypothesis.

$\underline{t = t_1 \mid t_2}$. Given some Σ in $\mathrm{FAIR}(t)$, let Σ' and Σ'' denote the left resp. right projections of Σ. By proposition 2.4, there exists some scheduling (f, g) of (Σ', Σ'') such that $\Sigma = \Sigma' {}_f\!\times_g \Sigma''$. Now, it is not difficult to see that $\mathrm{trace}(\Sigma)$ is the synchronous product $(\mathrm{trace}(\Sigma') \square f) \times (\mathrm{trace}(\Sigma'') \square g)$ and therefore belongs to $(\mathrm{trace}(\Sigma') \parallel \mathrm{trace}(\Sigma''))$. The desired result follows as above.

$\underline{t = t'[\pi]}$. Straightforward from the remark :

$\mathrm{Int}_R(p'[\pi]) = \emptyset$ iff $\mathrm{Int}_R(p') = \emptyset$.

$\underline{t = \langle \lambda \Rightarrow \lambda_1 : \lambda'_1, \dots, \lambda_n : \lambda'_n \rangle}$: immediate.

$\underline{t = t'[\lambda \triangleright \pi]}$. Consider some computation Σ in $\mathrm{FAIR}(t)$, let $\Sigma = (p_i, e_i)_{i \in \omega}$ with $p_0 = t$. We proceed by case analysis.

1 - $e_0 = \emptyset$, then $\mathrm{trace}(\Sigma) = \langle \{\lambda\}, \epsilon, \emptyset \rangle \cdot \langle \emptyset, \epsilon, \emptyset \rangle^\omega$ and $\varphi(\mathrm{trace}(\Sigma)) = \langle \{\lambda\}, \epsilon, \emptyset \rangle$.

2 - $e_0 \neq \emptyset$, then $e_0 = \{\lambda\}$ and $p_1 = (t' \mid t)[\pi]$. Let $\Sigma' = (p_i, e_i)_{i > 0}$ then :

$\mathrm{trace}(\Sigma) = \langle \emptyset, \lambda, \{\lambda\} \rangle \cdot \mathrm{trace}(\Sigma')$.

But Σ' is the π-image of some fair computation $\mathcal{C} \in \mathrm{FAIR}(t' \mid t)$, so that now :

$\mathrm{trace}(\Sigma') \in [\![\pi]\!] (\mathrm{trace}(\mathcal{C}_{\leftarrow}) \parallel \mathrm{trace}(\mathcal{C}_{\rightarrow}))$.

By proposition 2.2 and the induction hypothesis :

$\varphi(\text{trace}(\mathcal{C}_{\leftarrow})) \in \mathcal{M}(t')$.

Hence, by propositions 3.3 and 2.2 again :

$\varphi(\text{trace}(\mathcal{E})) \in (\lambda : \mathbb{I}\,\pi\,\mathbb{I}(\mathcal{M}(t') \,\|\, X))$

for $X = \varphi(\text{trace}(\text{FAIR}(t)))$.

Joining cases 1 and 2, one obtains $X \subseteq F(X)$ and thus $X \subseteq Y(F)$ for function F as in definition 3.8 □

proposition 3.5. $\forall t \in T_\Sigma$: $\mathcal{M}(t) \subseteq \text{Abs}(t)$.

proof By proposition 3.1, it is enough to show $\mathcal{M}(t) \subseteq \varphi(\text{trace}(\text{FAIR}(t)))$. Using implicitly the induction on Σ-terms, we establish this inclusion by case analysis according to the head symbol of t.

$t = \text{NIL}$: immediate.

$t = \langle \lambda_1, \dots, \lambda_n \rangle (t_1, \dots, t_n)$: straightforward by proposition 3.3.

$t = u \mid v$. Given h_t in $\mathcal{M}(t)$, one can find histories $h_u \in \mathcal{M}(u)$ and $h_v \in \mathcal{M}(v)$ such that $h_t \in h_u \,\|\, h_v$. By propositions 3.1 and 3.3 together with the induction hypothesis, $h_t = \varphi(T)$ for some $T \in U \,\|\, V$ where U and V are the respective traces of some open computations $\mathcal{E}' \in \text{FAIR}(u)$ and $\mathcal{E}'' \in \text{FAIR}(v)$, let $U = \langle \delta'_i, \rho'_i, d'_i \rangle_{i \in \omega}$ and $V = \langle \delta''_i, \rho''_i, d''_i \rangle_{i \in \omega}$. By definition 3.16, $T = ((U \square f) \times (V \square g))$ for some pair of monotonous onto functions f, g from ω to ω such that $(f(i) \neq f(i+1) \vee g(i) \neq g(i+1))$ for every i .

Since for $k>0$, $U(|\Sigma'|+k)$ and $V(|\Sigma''|+k)$ are both equal to $\langle \phi, \epsilon, \phi \rangle$ when these notations make sense, we can freely assume that f and g satisfy
$(f(i) \geq |\Sigma'| \Rightarrow f(i+1) = f(i)+1)$ and
$(g(i) \geq |\Sigma''| \Rightarrow g(i+1) = g(i)+1)$.
Nevertheless, (f,g) is not necessarily a scheduling of (Σ', Σ'') with respect to definition 2.5. Let (f', g') be the pair of functions from ω to ω such that :
$f'(j) =$ if $(j = i + \hat{f}(i))$ for some i then $f(i)$ else $f'(j+1)$,
$g'(j) =$ if $(j = i + \hat{g}(i))$ for some i then $g(i)$ else $g'(j+1)$,
where $\hat{f}$ and $\hat{g}$ are the <u>delay functions</u> given by :
$\hat{f}(i) = \text{card}\{j<i \mid (f(j)<|\Sigma'|) \& (g(j)<|\Sigma''|) \&$
$(f(j) \neq f(j+1)) \& (g(j) \neq g(j+1)) \& (\rho'_{f(j)} = \epsilon \vee \rho''_{g(j)} = \epsilon)\}$,
$\hat{g}(i) = \text{card}\{j \leq i \mid (f(j)<|\Sigma'|) \& (g(j)<|\Sigma''|) \&$
$(f(j) \neq f(j+1)) \& (g(j) \neq g(j+1)) \& (\rho'_{f(j)} = \epsilon \vee \rho''_{g(j)} = \epsilon)\}$.
Now, (f', g') is a scheduling of (Σ', Σ''), and it results from our construction that $\varphi(T)$ is equal to $\varphi((U \square f') \times (V \square g'))$, itself easily shown equal to $\varphi(\text{trace}(\Sigma'_{f' \times g'} \Sigma''))$. The desired conclusion $h_t \in \varphi(\text{trace}(\text{FAIR}(t)))$ follows by proposition 2.4.

<u>$t = t'[\pi]$</u> : straightforward.

<u>$t = \langle \lambda \Rightarrow \lambda_1 : \lambda'_1, \ldots, \lambda_n : \lambda'_n \rangle$</u> : immediate.

<u>$t = t'[\lambda \triangleright \pi]$</u>. Let function F as in definition 3.8, and consider h in $Y(F)$, the greatest fixpoint of F, one has to exhibit some computation $\mathcal{C}$ in $\text{FAIR}(t)$

such that $h = \varphi(\text{trace}(\mathcal{C}))$.

- Suppose that h belongs to the least fixpoint of F.

Since F is union continuous, although it is not continuous from above, the least fixpoint $y(F)$ of F may be written $F^{\omega}(\phi)$ or yet equivalently $\bigcup_i F^i(\phi)$.

Let i be the least integer such that $h \in F^i(\phi)$, and for that i, define programs p and p':

$p' = \langle\lambda\rangle_1((t' | \langle\lambda\rangle_2((t' | .. \langle\lambda\rangle_{i-1}((t' | \langle\lambda\rangle_i(\text{NIL}))[\pi])) .. [\pi]))[\pi])$,

$p = \langle\lambda\rangle_1((t' | \langle\lambda\rangle_2((t' | .. \langle\lambda\rangle_{i-1}((t' | t)[\pi])) .. [\pi]))[\pi])$.

By iterated application of the arguments developped for the first three cases already dealt with, it may be shown that there exists some computation $\mathcal{E}'$ in $\text{FAIR}(p')$, say $\mathcal{E}' = (p'_j, e'_j)_{j \in \omega}$, such that :

$h = \varphi(\text{trace}(\mathcal{E}'))$ and $(\forall j)(\rightarrow^{i-1} \underline{\text{out of}}\ e'_j)$

where we let R out of E iff $\forall \mu, \forall S : \mu_{RS} \notin e$.

Owing to the obvious similarity between p' and p, there also exists some computation $\mathcal{E}$ in $\text{FAIR}(p)$, say $\mathcal{E} = (p_j, e_j)_{j \in \omega}$, such that :

$h = \varphi(\text{trace}(\mathcal{E}))$ and $(\forall j)(\rightarrow^{i-1} \underline{\text{out of}}\ e_j)$.

By the rules of the axiomatic system $[\dot{\rightarrow}]$ given in section 2, the operational behaviours of t and $\langle\lambda\rangle((t' | t)[\pi])$ are identical up to the identification of states.

As a consequence, t and p are in a strong bisimulation, whence $\text{trace}(\mathcal{E}) = \text{trace}(\mathcal{C})$ for some $\mathcal{C} \in \text{FAIR}(t)$ and $h = \varphi(\text{trace}(\mathcal{C}))$.

- Suppose $h \notin Y(F)$. Then h belongs to $Y(F')$, the greatest fixpoint of $F'(X) = (\lambda : [\![\pi]\!](\mathcal{M}(t') \| X))$, considered as an equation over $\mathcal{P}(\mathcal{H})$. Set $h_0 = h$. From $Y(F') \subseteq Y(F)$, there exist sequences $(h_i)_{i\in\omega}$ and $(h'_i)_{i>0}$ of histories $h_i \in \mathcal{M}(t)$, $h'_i \in \mathcal{M}(t')$ such that for every $i > 0$:
$h_{i-1} \in (\lambda : [\![\pi]\!](h'_i \| h_i))$.
By the induction hypothesis, for every $i > 0$:
$h'_i = \varphi(\text{trace}(\Sigma'_i))$ for some $\Sigma'_i \in \text{FAIR}(t')$.
The remainder of the proof for that case is developped as a separate proposition (see 3.6 below) □

proposition 3.6. Given $t' \in T_\Sigma$ and Σ'_i in $\text{FAIR}(t')$, let $(h'_i)_{i>0}$ be a sequence of histories such that $\forall i$: $h'_i = \varphi(\text{trace}(\Sigma'_i))$. If $(h_i)_{i\in\omega}$ is a sequence of histories such that $\forall i > 0$: $h_{i-1} \in (\lambda : [\![\pi]\!](h'_i \| h_i))$, then $h_0 = \varphi(\text{trace}(\Sigma))$ for some $\Sigma \in \text{FAIR}(t)$, letting $t = t'[\lambda \triangleright \pi]$.

proof. Set $h_i = \langle \delta_i, \rho_i, d_i \rangle$ and $h'_i = \langle \delta'_i, \rho'_i, d'_i \rangle$, then $\rho_{i-1} \in (\lambda . \pi(\rho'_i \| \rho_i))$ for every $i > 0$. Let $\Sigma'_i = (t'_{ij}, e'_{ij})_{j\in\omega}$, then ρ'_i may be factored into $\eta(e'_{i0})\,\eta(e'_{i1}) \ldots \eta(e'_{ik}) \ldots$ - with $k \leq |\Sigma'_i|$ -.
Define $\dot{\eta} : (E_1 \cup E_2) \to \Lambda \cup \{.\}$:
$\dot{\eta}(e) =$ if $e \in E_1$ then $\eta(e)$ else $.$,
and let $\psi : (\Lambda \cup \{.\})^\infty \to \Lambda^\infty$ denote the $.$ erasing morphism.

Now, $\rho'_i = \psi(\dot\rho_i)$ for $\dot\rho_i$ equal to
$\dot\eta(e'_{i0})\,\dot\gamma(e'_{i1})\cdots\dot\eta(e'_{ik})\cdots$ - with $k \leq |\xi'_i|$ -,
and $(\rho'_i \| \rho_i) = \psi(\dot\rho_i \| \rho_i)$ - cf. definition 3.7 -.
Hence ρ_{i-1} is equal to $\lambda(\pi(\psi(\tilde\rho_i)))$ for some
$\tilde\rho_i$ in $(\dot\rho_i \| \rho_i)$, and there exist infinite factorizations

$\tilde\rho_i = \tilde\rho_{i1}\tilde\rho_{i2}\cdots\tilde\rho_{ik}\cdots$, $\tilde\rho_{ik} \in \Lambda \cup \{\epsilon, \cdot\}$

$\dot\rho_i = \dot\rho_{i1}\dot\rho_{i2}\cdots\dot\rho_{ik}\cdots$, $\dot\rho_{ik} \in \Lambda \cup \{\epsilon, \cdot\}$

$\rho_i = \rho_{i1}\rho_{i2}\cdots\rho_{ik}\cdots$, $\rho_{ik} \in \Lambda \cup \{\epsilon\}$

for which :

$(\dot\rho_{ik} \neq \epsilon \;\&\; \rho_{ik} \neq \epsilon) \Rightarrow ((\dot\rho_{ik} = \bar\rho_{ik}) \;\&\; (\tilde\rho_{ik} = \epsilon))$,

$(\dot\rho_{ik} = \epsilon \vee \rho_{ik} = \epsilon) \Rightarrow (\tilde\rho_{ik} = \dot\rho_{ik}\,\rho_{ik})$.

For $i>0$, these informations may be encoded into W_i,
the word in $(\{\downarrow\} \cup \{\leftarrow, \leftrightarrow, \rightarrow\})^\infty$ with occurences
$W_i(k)$, $0 \leq k \leq \sqcup\{j \mid (\dot\rho_{ij} \neq \epsilon \vee \rho_{ij} \neq \epsilon)\}$, as follows :
$W_i(k)$ = if $k=0$ then $\downarrow$ or else
if $((\dot\rho_{ik} \neq \epsilon) \;\&\; (\rho_{ik} \neq \epsilon))$ then $\leftrightarrow$ or else
if $(\rho_{ik} = \epsilon)$ then $\leftarrow$ or else $\rightarrow$.

Now define P, the set of pairs $\langle i,j\rangle$ for which
$W_i(j)$ is a defined occurence, ordered by
$\langle i,j\rangle \leq \langle k,l\rangle$ iff $(i=k \;\&\; j \leq l)$.
Clearly, this order is compatible with $\sim$, the least
equivalence over P generated from $\langle i,j\rangle \sim \langle k,l\rangle$ if
$(k=i+1) \;\&\; (W_i(j) \in \{\rightarrow, \leftrightarrow\}) \;\&\; (W_k(l) \neq \leftrightarrow) \;\&$
$(\mathrm{card}\{n \mid n \leq j \;\&\; W_i(n) \in \{\rightarrow, \leftrightarrow\}\} =$
$\mathrm{card}\{n \mid n \leq l \;\&\; W_k(l) \neq \leftrightarrow\})$.

Moreover, if m is the least integer for which π^{m-1} is a totally undefined function, then every $\sim$-class of P has at most $m+1$ elements. Thus, $(P \setminus \sim)_{\leqslant}$ is a countable semi-lattice such that every node in the associated Hasse diagram has its in/out degrees bounded by $m+1$. It is therefore possible to build an enumeration $\mathrm{enum}(P)$ of $P \setminus \sim$ along a refinement of the quotient order $\leqslant$. Henceforth, we take the convention that an equivalence class of P, say $\{\langle i_1, j_1\rangle, \ldots, \langle i_n, j_n\rangle\}$ with $i_1 < \ldots < i_n$, is denoted by the two-element set $\{\langle i_1, j_1\rangle, \langle i_n, j_n\rangle\}$ if $W_{i_1}(j_1) = \leftrightarrow$ or else by $\{\langle i_n, j_n\rangle\}$.

Let $\mathrm{enum}(P) = \chi_1 \chi_2 \ldots \chi_k \ldots$, where the χ_k are denotations of $\sim$-classes according to the above convention. If we set $\rho'_{ij} = \psi(\dot{\rho}_{ij})$ then ρ_0 may be factored into $\mu_1 \mu_2 \ldots \mu_j \ldots$ with the μ_j as follows:
$\mu_j =$ if $(\chi_j = \{\langle k, 0\rangle\})$ then $\pi^{k-1}(\lambda)$ or else if $(\chi_j = \{\langle k, i\rangle\}$ where $i > 0)$ then $\pi^k(\rho'_{ki})$ or else $\in$ for $\mathrm{card}(\chi_j) = 2$.
Now, $\mathrm{enum}(P)$ may be used to construct from $t_0 = t$ a computation $\mathcal{C} = (t_j, e_j)_{j \in \omega}$ as follows.
- Given t_j, define t_{j+1} as the program obtained by applying the following substitution S_{ki} for every element $\langle k, i\rangle$ in χ_{j+1} :

[if $i = 0$ then substitute $(t'|t)[\pi]$ for t in the position indicated by $\rightarrow^{k-1}$, or else substitute $t'_{k\,i}$ for $t'_{k\,i-1}$ in position $\rightarrow^{k-1}\leftarrow$,

- define e_j as the union of the following subsets $e''_{k\,i}$ for $\langle k, i\rangle$ in χ_{j+1} :

[$e''_{k\,i}$ = if $(i=0)$ then $\{\pi^{k-1}(\lambda) \rightarrow k-1\}$
or else $\rightarrow^{k-1}\leftarrow(\pi^k(e'_{k\,i-1}))$.

Then clearly, $\rho_0 = \eta((e_j)_{j\in\omega})$.

We shall now prove that $\mathcal{E}$ is a fair computation.

Suppose that condition 4 in proposition 2.1 is not satisfied for $\mathcal{E}$, and seek for a contradiction.

Now, for some R, j and γ :

$(\gamma \in Ult(\eta((e_i)_{i\in\omega}))) \;\&\; (\forall k \geq j)(\gamma \in Ext_R(t_k))$.

By the construction of $\mathcal{E}$ and the assumption $\varphi(trace(\mathcal{E}'_i)) = \langle \delta'_i, \rho'_i, d'_i\rangle$:

$R = (\rightarrow)^{p-1}\leftarrow S$ and $\gamma = \pi^p(\alpha)$ for some $\alpha \in \Lambda$ and $p > 0$ such that $\alpha \in \delta'_p$.

By the construction of $\mathcal{E}$ and the fact that π^m is totally undefined :

$\gamma = \pi^q(\beta)$ for some q such that $\beta \in Ult(\rho'_q)$.

Let us separate the various cases which arise from the comparison between p and q.

$\underline{p=q}$. Then $\langle \delta'_p, \rho'_p, d'_p\rangle$ does not belong to $\mathcal{H}$ by the injectivity of renamings - contradiction -

$q \leq p$: From the hypothesis $h_{i-1} \in (\lambda : [\![\pi]\!](h'_i \| h_i))$ and by the identity $h_i = \langle \delta_i, \rho_i, d_i \rangle$, $\beta = \pi^{p-q}(\alpha)$ implies $\beta \in \delta_q$, whence $\neg(h'_q \# h_q)$ – contradiction –

$p \leq q$: Either $\pi(\beta) \in Ult(\rho_{q-1})$ or $\bar{\beta} \in Ult(\rho_q)$ and then $\pi\{\beta, \bar{\beta}\} \subseteq d_{q-1}$. By definition 3.6, the following assertion is valid for every $i < q$:
$(\pi^i(\beta) \in Ult(\rho_{q-i})) \vee (\pi^i\{\beta, \bar{\beta}\} \subseteq d_{q-i})$.
In particular, it is valid for $i = q-p$, whence
$\neg(h'_p \# h_p)$ – contradiction –.

Suppose now that condition 3 in proposition 2.1 is not satisfied for Σ and seek for a contradiction.
By the assumption:
$(\exists j)(\forall k \geq j)(Int_R(t_k) \neq \emptyset)$ for some R.
By the construction of Σ, the hypothesis $\Sigma'_i \in FAIR(t')$ and the fact that π^m is totally undefined, there must exist $\alpha, \beta \in \Lambda$ and integers p, q such that:
$(p < q < p+m) \,\&\, (\alpha \in \delta'_p) \,\&\, (\alpha = \pi^{q-p}(\beta)) \,\&$
$(\bar{\beta} \in d'_q \cup \delta'_q)$.
If $\bar{\beta} \in \delta'_q$ then $\pi(\bar{\beta}) \in \delta_{q-1}$.
If $\bar{\beta} \in d'_q$ then $\pi(\bar{\beta}) \in d_{q-1}$ unless $\bar{\beta} \in \delta_q$ and hence $\pi(\bar{\beta}) \in \delta_{q-1}$.
By definition 3.6, the following assertion is valid for every $i < q$: $\pi^i(\bar{\beta}) \in (d_{q-i} \cup \delta_{q-i}) \cup \{\tau\}$.
Taking $i = q-p$, one obtains $\neg(h'_p \# h_p)$
– contradiction –.

To end the proof of proposition 3.6, the remains to show $h_0 = \varphi(\text{trace}(\mathcal{E}))$. Since $\mathcal{E} \in \text{FAIR}(t)$, this amounts to show $h_0 = \text{abs}(\mathcal{E})$. Now, by the construction of $\mathcal{E}$ and using the fact that π^m is the undefined function, $\text{abs}(\mathcal{E}) = \langle \delta, \rho_0, d \rangle$ with

- $\delta = \bigcup_{j=1}^{m-1} \pi^j(\delta'_j)$
- $d = (\bigcup_{j=1}^{m-1} \pi^j(d'_j)) \setminus \delta$.

Since $\delta_{j-1} = \pi(\delta'_j \cup \delta_j)$ for every $j > 0$, one gets :

$$\delta_0 = \bigcup_{j>0} \pi^j(\delta'_j) = \bigcup_{j=1}^{m-1} \pi^j(\delta'_j) = \delta .$$

Now, $d_{j-1} = \pi((d_j \setminus \delta'_j) \cup (d'_j \setminus \delta_j))$

$= \pi((d_j \cup d'_j) \setminus (\delta'_j \cup \delta_j))$

$= \pi(d_j \cup d'_j) \setminus \pi(\delta'_j \cup \delta_j)$ - injectivity of π -

$= \pi(d_j \cup d'_j) \setminus \delta_{j-1}$.

By the obvious induction, one gets :

$$d_0 = \bigcup_{j>0} (\pi^j(d'_j) \setminus \delta_0) = \bigcup_{j>0} \pi^j(d'_j) \setminus \delta = d$$

<u>end of the proof</u>.

4. A FULLY OBSERVATIONAL MODEL.

We intend presently to turn the material of section 3 into a significantly less discriminative model. Whereas the operational model reflects the computation histories of programs, the observational model reflects their co-computations. Our first task is to make this connection precise.

4.1. THE OBSERVATIONAL SETTING

definition 4.1. Let $p \in T_\Sigma$, the large set of observations OBS(p) of p is the following subset of $\mathcal{H}$:

$$OBS(p) = \{ abs(\mathcal{C}_\rightarrow) \mid \mathcal{C} \in fair(p|q) \text{ for some } q \}.$$

Before we proceed with the technical development, we argue briefly on the fact that the term "observations" is not misused: (We leave it to later stages to show that the associated concept is the right one for asynchrony). Now, the history of an open computation in FAIR(q) does only depend on a derivation sequence such as $(q_i \xrightarrow{e_i} q_{i+1})_{i \in I}$. Thus, for a closed computation $\mathcal{C}$ in $fair(p|q)$, $abs(\mathcal{C}_\rightarrow)$ does not encompass any form of inquiry into the states or state changes

of p. Nevertheless, the observational abstraction of p depends in a precise way upon its operational abstraction, as shown by the following ...

proposition 4.1. $OBS(p)$ is the set made out of the histories $\langle \delta, \rho, d \rangle$ in $\mathcal{H}$ which are compatible with some history $\langle \delta', \rho', d' \rangle$ in $Abs(p)$ such that $\rho = \bar{\rho}'$.

proof It is a straightforward consequence of propositions 2.2, 2.3 and 3.1, 3.2 taken together □

Not surprisingly, $OBS(p)$ contains a lot of redundant elements which can be dropped without any loss of information. Part of them may be eliminated by chosing instead of OBS the following variant Obs.

definition 4.2. Let $p \in T_\Sigma$ then $obsort(p)$, the observable sort of p, is the minimal sort L such that $\rho \in L^\infty$ for every $\langle \delta, \rho, d \rangle$ in $OBS(p)$, and $Obs(p)$, the small set of observations of p, is the intersection $OBS(p) \cap \mathcal{H}_L$ for $L = obsort(p)$.

proposition 4.2. If L is the observable sort of p, then $OBS(p) = \{ \langle \delta, \rho, d \rangle \in \mathcal{H} \mid \langle \delta \cap L, \rho, d \cap L \rangle \in Obs(p) \}$.

proof. $Obs(p) \subseteq OBS(p)$ obviously. Now, $Abs(p)$ is included in $\mathcal{H}_L$ by the first corollary of proposition 3.1 and the above proposition 4.1. Therefore, for any $\langle \delta', \rho', d' \rangle$ in $Abs(p)$ and for every $\langle \delta, \rho, d \rangle$ in $\mathcal{H}$: $\langle \delta, \rho, d \rangle \# \langle \delta', \rho', d' \rangle$ iff $\langle \delta \cap L, \rho, d \cap L \rangle \# \langle \delta', \rho', d' \rangle$ □

By proposition 4.2, Obs(p) is a faithfull representation of OBS(p). Next subsection suggests a faithfull representation of Obs in terms of Abs.

4.2. A CONNECTION BETWEEN Obs and Abs.

Recalling the first corollary of proposition 3.1, it is clear from definitions 3.3 and 3.6 that small sets of observations may be characterized as follows: let $L = obsort(p)$ then Obs(p) is the set of the histories $\langle \delta, p, d \rangle$ for which one can find some history $\langle \delta', p', d' \rangle$ in Abs(p) such that $(p = \bar{p}') \& (\delta \subseteq L \setminus (\bar{d}' \cup \bar{\delta}')) \& ((d \cup \delta) \subseteq L \setminus \bar{\delta}')$. This remark suggests to make $\mathcal{H}$ into an ordered structure as follows.

<u>definition 4.3</u>. $\leqslant$ is the <u>order relation on $\mathcal{H}$</u> such that $\langle \delta, p, d \rangle \leqslant \langle \delta', p', d' \rangle$ iff $(p = p') \& (\delta \subseteq \delta') \& ((d \cup \delta) \subseteq (d' \cup \delta'))$.
We let $\hat{}$ (resp. $\check{}$) denote the operators upon $\mathcal{P}(\mathcal{H})$ which send parts of $\mathcal{H}$ to their greatest subsets of maximal (resp. minimal) elements.

Obviously, small sets of observations are downwards closed w.r.t. $\leqslant$, whereby $\widehat{Obs}(p)$ is a faithfull representation of Obs(p). In addition,

$\widehat{Obs}(p)$ is fully determined by $\check{Abs}(p)$: evidence of this fact is given by the remark above. After some notations are introduced, we state the exact dependance of $\widehat{Obs}$ upon $\check{Abs}$.

definition 4.4. For $L \in Sorts$, let $IO_L : \mathcal{H}_L \to \mathcal{P}(\mathcal{H}_L)$:

$IO_L(<\delta', \rho', d'>) = \{ <\delta, \rho, d> \in \mathcal{H} \mid$

$(\rho = \bar{\rho}') \& (\delta \subseteq L \setminus (\bar{d}' \cup \bar{\delta}')) \& ((d \cup \delta) \subseteq L \setminus \bar{\delta}') \}$.

To each sort L we associate the operators IO_L and OI_L from $\mathcal{P}(\mathcal{H}_L)$ to $\mathcal{P}(\mathcal{H}_L)$ such that :

$IO_L(P) = \bigcup_h IO_L(h)$, $h \in P$

$OI_L(P) = \{ h \in \mathcal{H}_L \mid (\forall h' \in IO_L(h))(\exists h'' \in P)(h' \leqslant h'') \}$.

proposition 4.3. given p in T_Σ, let L be the minimal sort which satisfies $\check{Abs}(p) \subseteq \mathcal{H}_L$, then

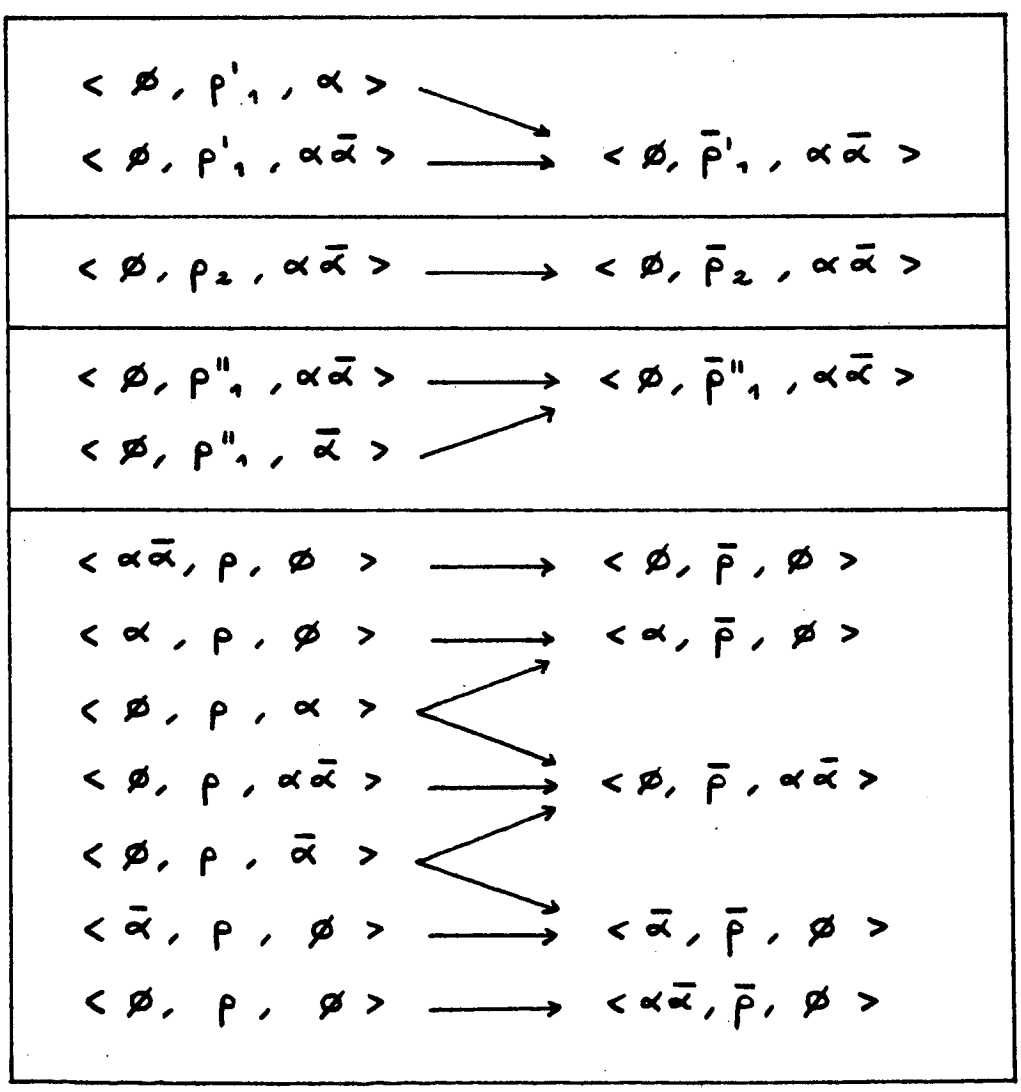
$\widehat{Obs}(p) = \widehat{IO_L}(Abs(p)) = \widehat{IO_L}(\check{Abs}(p))$.

proof. immediate by corollary 1 of proposition 3.1 □

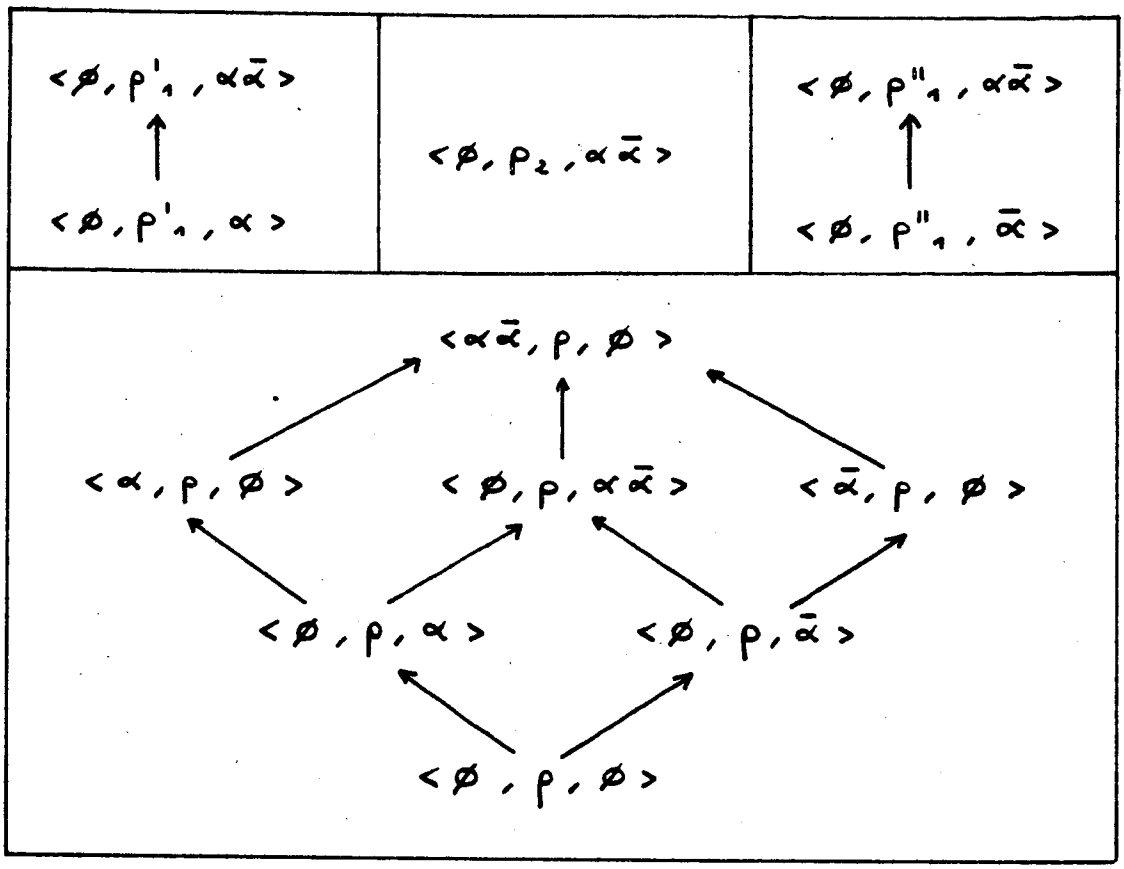
We are now in a position where we may ask ourselves wether $\check{Abs}(p)$ is in turn fully determined by $\widehat{Obs}(p)$. As might be guessed, this is not the case. In order to understand what happens, consider for instance the histories $h' = <\emptyset, \alpha^\omega, \alpha>$ and $h'' = <\emptyset, \alpha^\omega, \alpha\bar{\alpha}>$ under the assumption $L = \{\alpha, \bar{\alpha}\}$, then $\check{OI}_L(IO_L(\{h''\})) = \{h'\}$. Also consider the histories $h_1 = <\alpha, \epsilon, \emptyset>$, $h_2 = <\emptyset, \epsilon, \alpha>$ and $h_3 = <\emptyset, \epsilon, \alpha\bar{\alpha}>$, then $\check{OI}_L(IO_L(\{h_1, h_3\})) = \{h_2\}$.

An adequate normalization of parts of $\mathcal{HC}_L$ will cope with these two categories of confusion cases . Some diagrams may help to apprehend the principle of the normalization. Figure 1 gives the graph of $\widehat{IO}_L$ for $L = \{\alpha, \bar{\alpha}\}$, and figure 2 is the Hasse diagram of $\leqslant$ on $\mathcal{HC}_L$. In both figures $\rho, \rho'_1, \rho''_1, \rho_2$ denote any words in L^∞ with respective sets of ultimate letters $\emptyset, \{\alpha\}, \{\bar{\alpha}\}, \{\alpha, \bar{\alpha}\}$.

- Figure 1 -



- Figure 2 -



definition 4.5. For $\alpha \in \Delta$, we define $norm_\alpha$:

$\mathcal{HC} \to \mathcal{P}(\mathcal{HC})$: $norm_\alpha(\langle \delta, \rho, d \rangle) =$

if $(Ult(\rho) \cap \{\alpha, \bar{\alpha}\} \neq \emptyset)$ *then* $\{\langle \delta, \rho, d \cup \{\alpha, \bar{\alpha}\} \rangle\}$ *or else*

if $(\{\alpha, \bar{\alpha}\} \subseteq d \vee \{\alpha, \bar{\alpha}\} \cap d = \emptyset)$ *then* $\{\langle \delta, \rho, d \rangle\}$ *or else*

if $(\alpha \in d)$ *then* $\{\langle \delta \cup \{\alpha\}, \rho, d \setminus \{\alpha\} \rangle, \langle \delta, \rho, d \cup \{\bar{\alpha}\} \rangle\}$ *or else*

if $(\bar{\alpha} \in d)$ *then* $\{\langle \delta \cup \{\bar{\alpha}\}, \rho, d \setminus \{\bar{\alpha}\} \rangle, \langle \delta, \rho, d \cup \{\alpha\} \rangle\}$.

These operators are extended to work on sets according to

$norm_\alpha(P) = \bigcup_h norm_\alpha(h)$, $h \in P$.

For $L \in Sorts$, *we let* $norm_L : \mathcal{P}(\mathcal{HC}_L) \to \mathcal{P}(\mathcal{HC}_L)$:

$norm_L = norm_{\alpha_1} \circ \ldots \circ norm_{\alpha_n}$ *if* $\Delta \cap L = \{\alpha_1 \ldots \alpha_n\}$.

definition 4.5 continued. Generalizing the above, norm is the operator on $\bigcup_L \mathcal{P}(\mathcal{H}_L)$ such that $norm(P) = norm_L(P)$ for $P \subseteq \mathcal{H}_L$. A subset P of $\mathcal{H}$ is *normalized* iff $P = norm(P)$. A subset P of $\mathcal{H}$ is *unambiguous* iff it is normalized and does not contain any two different comparable histories. $\mathcal{K}$ is the set of the unambiguous subsets of $\bigcup_L \mathcal{P}(\mathcal{H}_L)$.

Clearly, for any $P \subseteq \mathcal{H}_L$, $\widehat{IO}_L(P)$ is unambiguous and moreover equal to $\widehat{IO}_L(\check{norm}(P))$ – see figure 1 –. In particular, for any program p with observable sort L, $\widehat{Obs}(p) = \widehat{IO}_L(\check{norm}(Abs(p)))$. We are willing to show that for p ranging over T_Σ, $\check{norm} \circ Abs(p)$ is a faithfull representation of $\widehat{Obs}(p)$ or of $Obs(p)$ or of $OBS(p)$. For that purpose, it is enough to prove $OBS(p) \subseteq OBS(q)$ iff $\check{norm} \circ Abs(p) \sqsubset_I \check{norm} \circ Abs(q)$ where $\sqsubset_I$ is the order on $\mathcal{K}$ given in the following ...

definition 4.6 $\underset{\sim}{\sqsubseteq}_I$ is the preorder on $\bigcup_L \mathcal{P}(\mathcal{H}_L)$ such that $P \underset{\sim}{\sqsubseteq}_I Q$ iff $(\forall h \in P)(\exists h' \in Q)(h' \leqslant h)$, and $\underset{\sim}{\sqsubseteq}_O$ is the preorder on $\bigcup_L \mathcal{P}(\mathcal{H}_L)$ such that $P \underset{\sim}{\sqsubseteq}_O Q$ iff $(\forall h \in P)(\exists h' \in Q)(h \leqslant h')$. $\sqsubset_I$ and $\sqsubset_O$ are the restrictions on $\mathcal{K}$ of $\underset{\sim}{\sqsubseteq}_I$ resp. $\underset{\sim}{\sqsubseteq}_O$.

lemma 4.1 For P, Q ranging over $\mathcal{K} \cap \mathcal{P}(\mathcal{H}_L)$, $\widehat{IO}_L(P) \sqsubset_O \widehat{IO}_L(Q)$ iff $P \sqsubset_I Q$.

proof: easily checked from figures 1 and 2 □

proposition 4.4. For p, q ranging over T_Σ :
$OBS(p) \subseteq OBS(q)$ *iff* $\check{norm} \circ Abs(p) \sqsubset_I \check{norm} \circ Abs(q)$.

proof. Before we establish the reciprocal implications, we set down $L' = obsort(p)$, $L = obsort(q)$, and $\nabla = (L \setminus L') \cup (L' \setminus L)$.

$\Rightarrow$: Assume $OBS(p) \subseteq OBS(q)$.
Then $L' \subseteq L$ and $\widehat{OBS}_L(p) \underset{\sim}{\sqsubset}_0 \widehat{OBS}_L(q)$, letting $OBS_L(t) = OBS(t) \cap \mathcal{H}_L$. By proposition 4.2 :
$\widehat{OBS}_L(p) = \{ \langle \delta \cup \nabla, \rho, d \rangle \mid \langle \delta, \rho, d \rangle \in \widehat{Obs}(p) \}$,
but $\widehat{Obs}(p) = \widehat{IO}_{L'}(\check{norm}(Abs(p)))$ by proposition 4.3 , whence $\widehat{OBS}_L(p) = \widehat{IO}_L(\check{norm}(Abs(p)))$ is clear from definition 4.4 . By proposition 4.3 again :
$\widehat{IO}_L(\check{norm}(Abs(p))) \sqsubset_0 \widehat{IO}_L(\check{norm}(Abs(q)))$,
and the result follows by lemma 4.1.

$\Leftarrow$: Assume $\check{norm} \circ Abs(p) \sqsubset_I \check{norm} \circ Abs(q)$.
Then $L' \subseteq L$ by proposition 4.1 . By lemma 4.1 :
$\widehat{IO}_L(\check{norm}(Abs(p))) \sqsubset_0 \widehat{IO}_L(\check{norm}(Abs(q)))$.
Hence $\widehat{OBS}_L(p) \underset{\sim}{\sqsubset}_0 \widehat{OBS}_L(q)$ with the same notations as above , and finally $OBS(p) \subseteq OBS(q)$ by proposition 4.2 $\square$

4.3. TOWARDS AN OBSERVATIONAL MODEL

Let the *observational preorder* $\lesssim$ on T_Σ :
$p \lesssim q$ *iff* $OBS(p) \subseteq OBS(q)$. The intuition behind this definition is as follows :

p *implements* q iff for any program r, no computation of r in $(p|r)$ allows to recognize that q has been replaced by p. (By proposition 2.3, this interpretation is consistant with definitions 3.1 and 4.1).

Our intend is to obtain a fully abstract model of T_Σ w.r.t. the observational preorder $\lesssim$. From proposition 4.4, an economical way to get such a model is to derive it from $\mathcal{M}$, for $\mathcal{M}$ and Abs have been proven identical in section 3. More precisely, let $\mathcal{N} = \check{norm} \circ \mathcal{M}$, then by proposition 4.4: $p \lesssim q$ iff $\mathcal{N}(p) \sqsubset_I \mathcal{N}(q)$.

Now, all the problem is to show that $\mathcal{N}$ may be characterized by equations derived from the defining equations of $\mathcal{M}$. In section 3.3, $\mathcal{M}$ has been defined by six equations of the generic form $\mathcal{M}(op(\vec{t_i})) = OP(\overrightarrow{\mathcal{M}(t_i)})$, where $op \in \Sigma$ and the OP are operators on $\mathcal{P}(\mathcal{H})$ with corresponding arities. We will show that $\mathcal{N}$ may also be characterized by six equations of the form $\mathcal{N}(op(\vec{t_i})) = \widetilde{OP}(\overrightarrow{\mathcal{N}(t_i)})$, where the $\widetilde{OP}$ are the operators on $\mathcal{K}$ given by: $\widetilde{OP}(Q) = \check{norm} \circ OP(Q)$. (Notice the identity $\check{norm} = \check{} \circ norm = norm \circ \check{}$). The proofs of the propositions are exiled to division 4.5.

proposition 4.5 For any $op \in \Sigma_n$, $n>0$, and for every sets of histories $P_i \subseteq \bigcup_L \mathcal{P}(\mathcal{H}_L)$:
$\check{OP}(P_1,\dots,P_n) = \check{OP}(\check{P}_1,\dots,\check{P}_n)$. □

proposition 4.6 For any $op \in \Sigma_n$, $n>0$, and for every sets of histories $P_i \subseteq \bigcup_L \mathcal{P}(\mathcal{H}_L)$:
$norm \circ OP(P_1,\dots,P_n) = OP(norm(P_1),\dots,norm(P_n))$ □

Essential to the proof of the above proposition is ...

lemma 4.2 For every $\alpha \in \Lambda$ and for every $h', h'' \in \mathcal{H}$:
$norm_\alpha(h' \| h'') = norm_\alpha(h') \| norm_\alpha(h'')$ □

Straightforward consequences of propositions 4.5 and 4.6 are the following lemmas 4.3 and 4.4 which show how the equations of $\mathcal{N}$ may be derived from the equations of $\mathcal{M}$.

lemma 4.3 For any $op \in \Sigma_n$, $n>0$, and for every sets of histories $P_i \subseteq \bigcup_L \mathcal{P}(\mathcal{H}_L)$:
$\check{norm} \circ OP(P_1,\dots,P_n) = \check{OP}(\check{norm}(P_1),\dots,\check{norm}(P_n))$.

proof. $\check{norm} \circ OP(P_1,\dots,P_n) = norm \circ \check{OP}(P_1,\dots,P_n)$
$= norm \circ \check{OP}(\check{P}_1,\dots,\check{P}_n) = \check{} \circ norm \circ OP(\check{P}_1,\dots,\check{P}_n)$
$= \check{OP}(norm(\check{P}_1),\dots,norm(\check{P}_n))$
$= \check{OP}(\check{norm}(P_1),\dots,\check{norm}(P_n))$ □

lemma 4.4 For any op in Σ_n, $n>0$:
$\mathcal{N}(op(\vec{t_i})) = \check{OP}(\overrightarrow{\mathcal{N}(t_i)})$

proof $\mathcal{N}(op(\vec{t_i})) = \check{norm} \circ \mathcal{M}(op(\vec{t_i}))$
$= \check{norm} \circ OP(\overrightarrow{\mathcal{M}(t_i)}) = \check{OP} \circ norm(\overrightarrow{\mathcal{M}(t_i)})$
$= \check{OP}(\check{norm}(\overrightarrow{\mathcal{M}(t_i)})) = \check{OP}(\overrightarrow{\mathcal{N}(t_i)})$ □

Additional informations are given by the following variant of proposition 4.5 :

lemma 4.5. For any $op \in \Sigma_n$, $n>0$, the associated operator $\check{OP} : K^n \to K$ is monotonous w.r.t. $\sqsubset_I$.

proof. Notice first that the above statement makes sense, since $\check{OP} : K^n \to K$ is well defined by lemma 4.3. The remainder of the proof is a straightforward adaptation of the arguments given for proposition 4.5 □

4.4. THE OBSERVATIONAL MODEL

Let $[\mathcal{M}(op_j(\vec{t_i})) = OP_j(\overrightarrow{\mathcal{M}(t_i)})]_{j=1\ldots6}$ denote the defining equations of the meaning function $\mathcal{M}$. From now on, we let $\mathcal{N} : T_\Sigma \to K$ denote the observational meaning function inductively given by $[\mathcal{N}(op_j(\vec{t_i})) = \widetilde{OP}_j(\overrightarrow{\mathcal{N}(t_i)})]_{j=1\ldots6}$ where $\widetilde{OP}_j = \check{norm} \circ OP_j$ if $op_j \in \Sigma_0$ or $\check{OP}_j$ otherwise.

theorem 4.1. $\mathcal{N}$ is a *fully abstract* model of T_Σ w.r.t. the observational preorder $\lesssim : \mathcal{N}(t) \sqsubset_I \mathcal{N}(t')$ iff ($OBS(\mathcal{C}(t)) \subseteq OBS(\mathcal{C}(t'))$ holds for every Σ-context $\mathcal{C}$). Moreover, the following equivalence

holds : $\mathcal{N}(t) \sqsubset_I \mathcal{N}(t')$ iff $OBS(t) \subseteq OBS(t')$.

proof. The latter equivalence results from proposition 4.4 and theorem 3.1 . Owing to the induction on Σ-contexts , the former equivalence follows from the latter by lemmas 4.4 and 4.5 □

Told in other words, $\mathcal{N}(p) \sqsubset_I \mathcal{N}(q)$ iff for every Σ-context $\mathcal{C}$ and for every Σ-program τ, no computation of τ in $(\mathcal{C}(p) \mid \tau)$ allows to recognize that q has been replaced by p. To complete the above theorem, it may be verified for instance that $\mathcal{N}((\langle\alpha\rangle(p) \mid \langle\bar{\alpha}\rangle(NIL))[/\alpha])$ equals $\mathcal{N}(p)$ for every $p \in T_\Sigma$ and $\alpha \notin sort(p)$ - we leave this exercise to the reader - . Hence, we can claim that $\mathcal{N}$ is a *fully abstract model of fair asynchrony*. Notice that some of the properties of $\mathcal{M}$ are directly inherited by $\mathcal{N}$. For instance, $\mathcal{N}(t)$ differs from $\emptyset$ for every $t \in T_\Sigma$, and if $\langle\delta, p, d\rangle$ belongs to $\mathcal{N}(t)$ then some history $\langle\delta', p', d'\rangle$ occurs in $\mathcal{N}(t)$ for each proper left factor p' of p.

We have still to discuss in what extent this model is of practical interest for proving the semantic equivalence of Σ-programs. In that respect, the provision of some induction rule appears critical. Next theorem tolds us that greatest fixpoint induction is valid.

theorem 4.2. $\mathcal{N}$ *is a greatest fixpoint semantics.*

proof We have to show that $\mathcal{N}(t[\lambda \triangleright \pi])$ is the greatest fixpoint (w.r.t. $\sqsubset_I$) of some equation over $\mathcal{K}$ in which $\mathcal{N}(t)$ appears as a constant. So, let $Q = \mathcal{N}(t)$ and define $F : \mathcal{P}(\mathcal{H}) \to \mathcal{P}(\mathcal{H})$:
$F(X) = \langle \lambda, \epsilon, \emptyset \rangle + (\lambda : [\![\pi]\!](Q \parallel X))$. By lemma 4.4:
$\mathcal{N}(t[\lambda \triangleright \pi]) = \vee(Y(F))$. Now, $Y(F) = F^{\alpha}(\mathcal{H})$ for any sufficiently high ordinal α, say $\alpha > \omega$.
By proposition 4.5, $\check{F}(F^{\alpha}(\mathcal{H})) = \check{F}(\vee_0 F^{\alpha}(\mathcal{H}))$.
Hence, $\mathcal{N}(t[\lambda \triangleright \pi]) = \check{F}(\check{F}(F^{\alpha}(\mathcal{H}))) = \check{F}(\mathcal{N}(t[\lambda \triangleright \pi]))$
and $\mathcal{N}(t[\lambda \triangleright \pi])$ is a fixpoint of $\check{F} : \mathcal{K} \to \mathcal{K}$.
To show that it is a greatest fixpoint, consider Z in $\mathcal{K}$ and suppose $Z = \check{F}(Z)$. Since $F(Z) \subseteq F(F(Z))$, $F(Z)$ is a pre-fixpoint of Z and thus $F(Z) \subseteq Y(F)$.
Clearly, this implies in turn $\check{F}(Z) \sqsubset_I \mathcal{N}(t[\lambda \triangleright \pi])$. □

This theorem taken alone does not indicate clearly a proof system for $(T_\Sigma, \lesssim)$, because the major difficulties are now concentrated on the fair composition of infinitary languages. However, let Σ' be the signature obtained from Σ when deleting iterators $[\lambda \triangleright \pi]$, then a decision procedure is available for $(T_{\Sigma'}, \lesssim)$, for in that restricted case $\mathcal{N}(t)$ is an effectively given rational language. More details will appear in section 5.

To sum up, we finally include the <u>characteristic equations of $\mathcal{N}$</u>.

$$\mathcal{N}(\mathrm{NIL}) = \langle \emptyset, \epsilon, \emptyset \rangle$$

$$\mathcal{N}(\langle \lambda_1, \ldots, \lambda_n \rangle (t_1, \ldots, t_n)) =$$
$$\langle \{\lambda_1 \ldots \lambda_n\}, \epsilon, \emptyset \rangle + \check{\vee}\left(\sum_{i=1}^{n} \lambda_i : \mathcal{N}(t_i)\right)$$

$$\mathcal{N}(t_1 \mid t_2) = \check{\vee}(\mathcal{N}(t_1) \parallel \mathcal{N}(t_2))$$

$$\mathcal{N}(t\,[\pi]) = \check{\vee}(\mathbb{I}\pi\mathbb{I}\,\mathcal{N}(t))$$

$$\mathcal{N}(t\,[\lambda \triangleright \pi]) = Y(\check{F}) \quad \text{where}$$
$$F(X) = \langle \lambda, \epsilon, \emptyset \rangle + (\lambda : \mathbb{I}\pi\mathbb{I}(\mathcal{N}(t) \parallel X)))$$

$$\mathcal{N}(\langle \lambda \Rightarrow \lambda_1 : \lambda'_1, \ldots, \lambda_n : \lambda'_n \rangle) =$$
$$\check{\mathrm{norm}}(\mathcal{M}(\langle \lambda \Rightarrow \lambda_1 : \lambda'_1, \ldots, \lambda_n : \lambda'_n \rangle))$$

4.5. PROOFS OF PROPOSITIONS

<u>proof of proposition 4.5</u>.

Let $\underset{\sim}{\sqsubseteq}$ denote the preorder on $\bigcup_L \mathcal{P}(\mathcal{H}_L)$ such that $P \underset{\sim}{\sqsubseteq} Q$ iff $(P \underset{\sim}{\sqsubseteq}_0 Q \;\&\; Q \underset{\sim}{\sqsubseteq}_I P)$, or equivalently: $((\forall h \in P, \exists h' \in Q : h \lesssim h') \;\&\; (\forall h' \in Q, \exists h \in P : h \lesssim h')$. For some of the operators $op \in \Sigma$, the restriction of OP to $\bigcup_L \mathcal{P}(\mathcal{H}_L)$ is monotonous w.r.t $\underset{\sim}{\sqsubseteq}$. This property is clearly true of the operators associated with symbols $\langle \lambda_1, \ldots, \lambda_n \rangle$ and $[\pi]$, but it does not hold for parallel composition $\mid$ and iterators $[\lambda \triangleright \pi]$.

As regards parallel composition, it is enough to prove $(\{h'\} \| \{h\}) \sqsubseteq (\{h', h''\} \| \{h\})$ for $h' \leq h''$, and this fact is obvious from definitions 3.6 and 4.3 :
- if $\neg(h'' \# h)$ the two members are equal,
- if $(h'' \# h)$ then necessarily $h' \# h$ and the desired conclusion follows.

Let us now turn to iterators such as $[\lambda \triangleright \pi]$. Set $OP(Q) = Y_X(F)$ for function F given by : $F(Q, X) = \langle \lambda, \epsilon, \phi \rangle + (\lambda : [\![\pi]\!](Q \| X))$, and also define $G(Q, X) = (\lambda : [\![\pi]\!](Q \| X))$. Recall that $Y_X(F)$ is the union $y_X(F) \cup Y_X(G)$ of the least fixpoint of F and the greatest fixpoint of G : if we let $F'(Q) = y_X(F)$ and $G'(Q) = Y_X(G)$ then we may satisfy ourselves with proving :
$F'(\check{Q}) \sqsubseteq F'(Q)$ and $G'(\check{Q}) \sqsubseteq G'(Q)$.

Now, $F'(Q) = \bigcup_{i \geq 0} ((F''_Q)^i(\phi))$ for function $F''_Q(P) = F(Q, P)$, and $(F''_{\check{Q}})^i(\phi) \sqsubseteq (F''_Q)^i(\phi)$ by the induction on i, whence $F'(\check{Q}) \sqsubseteq F'(Q)$.

We finally prove $G'(\check{Q}) \sqsubseteq G'(Q)$. Given some history h''_0 in $G'(Q)$, we shall exhibit a corresponding h'_0 in $G'(\check{Q})$ such that $h'_0 \leq h''_0$. So, let m be the least integer for which π^m is the undefined function. By the definition of G', there exist a sequence $(h''_i)_{i \in \omega}$

of histories in $\mathcal{H}$ and a sequence $(h_i)_{i>0}$ of histories in Q such that $\forall i : h''_i \in (\lambda : [\![\pi]\!](h_{i+1} \| h''_{i+1}))$. For every i, chose $\underline{h}_i = \langle \underline{\delta}_i, \rho_i, \underline{d}_i \rangle$ in $\check{Q}$ such that $\underline{h}_i \leqslant h_i$, and define $h'_i = \langle \delta'_i, \rho''_i, d'_i \rangle$, with ρ''_i taken from h''_i and δ'_i, d'_i as follows :
$\delta'_i = \bigcup_{j=1}^{m-1} \pi^j(\underline{\delta}_{j+i})$, $d'_i = (\bigcup_{j=1}^{m-1} \pi^j(\underline{d}_{j+i})) \setminus \delta'_i$.
Clearly, $h'_i \in (\lambda : [\![\pi]\!](\underline{h}_{i+1} \| h'_{i+1}))$ for every i, and hence $h'_0 \in G'(\check{Q})$. Now, set $h_i = \langle \delta_i, \rho_i, d_i \rangle$ then $\delta''_0 = \bigcup_{j=1}^{m-1} \pi^j(\delta_{j+i})$ and $d''_0 = (\bigcup_{j=1}^{m-1} \pi^j(d_{j+i})) \setminus \delta''_0$, whence $h'_0 \leqslant h''_0$ follows by the assumption $\underline{h}_i \leqslant h_i$.

proof of proposition 4.6.

The property to be demonstrated is clearly true of the operators associated with symbols $\langle \lambda_1 \dots \lambda_n \rangle$ and $[\pi]$. For $op = 1$, it is a straightforward consequence of lemma 4.2 proven below. So, there remains to consider the case of iterators $[\lambda \triangleright \pi]$.

Define F, G, F', G', F''_Q as in the proof of the above proposition 4.5. We have to show the equality between sets $\mathrm{norm}(F'(Q)) \cup \mathrm{norm}(G'(Q))$ on one side and $F'(\mathrm{norm}(Q)) \cup G'(\mathrm{norm}(Q))$ on the other side. By the induction on i : $\mathrm{norm}((F''_Q)^i(\emptyset)) = F''^{\,i}_{\mathrm{norm}(Q)}(\emptyset)$, whence clearly $\mathrm{norm}(F'(Q)) = F'(\mathrm{norm}(Q))$. We proceed with proving $\mathrm{norm}(G'(Q)) = G'(\mathrm{norm}(Q))$ by reciprocal inclusions.

norm (G'(Q)) ⊆ G'(norm (Q)).

Let $h'_0 \in$ norm (G'(Q)), then $h'_0 \in$ norm (h''_0) for some h''_0 in G'(Q). Let sequences $(h''_i)_{i\in\omega}$ and $(h_i)_{i>0}$ be determined from h''_0 as in proposition 4.5. Owing to the already established identity norm ($\lambda : [\![\pi]\!] (h' \| h'')$) = ($\lambda : [\![\pi]\!]$ (norm(h') ‖ norm(h''))), there exist corresponding sequences $(h'_i)_{i\in\omega}$ and $(\underline{h}_i)_{i>0}$ such that $\forall i$:

$(h'_i \in$ norm $(h''_i)) \;\&\; (\underline{h}_{i+1} \in$ norm $(h_{i+1})) \;\&$

$(h'_i \in (\lambda : [\![\pi]\!] (\underline{h}_{i+1} \| h'_{i+1})))$.

Now, $(\forall i)(h_{i+1} \in Q)$ implies $(\forall i)(h'_i \in G'(\text{norm}(Q)))$ by the definition of G', hence $h'_0 \in G'$(norm(Q)).

G'(norm(Q)) ⊆ norm (G'(Q)).

Let $h'_0 \in G'$(norm(Q)). By the definition of G', there exist sequences $(h'_i)_{i\in\omega}$ in $\mathcal{H}^\omega$ and $(\underline{h}_i)_{i>0}$ in (norm(Q))$^\omega$ such that $\forall i$:

$h'_i \in (\lambda : [\![\pi]\!] (\underline{h}_{i+1} \| h'_{i+1}))$.

For strictly positive i, chose h_i in Q such that $\underline{h}_i \in$ norm(h_i), then $h_i \leqslant \underline{h}_i$ by the definition of normalization. Set down:

$h_i = \langle \delta_i, \rho_i, d_i \rangle$, $\underline{h}_i = \langle \underline{\delta}_i, \rho_i, \underline{d}_i \rangle$, $h'_i = \langle \delta'_i, \rho'_i, d'_i \rangle$.

Now define $h''_i = \langle \delta''_i, \rho'_i, d''_i \rangle$, where δ''_i and d''_i are expressed as follows for $\pi^m = \tau$:

$\delta''_i = \bigcup_{j=1}^{m-1} \pi^j (\delta_{j+i})$, $d''_i = (\bigcup_{j=1}^{m-1} \pi^j (d_{j+i})) \setminus \delta''_i$.

§ We show that the h''_i belong to $\mathcal{HC}$.

Suppose $d''_i \cap \bar{\delta}''_i \neq \emptyset$ for some i, then one can find $\alpha, \zeta \in \Lambda$, $j \in \omega$ and $k < m$ such that $\alpha = \pi^k(\zeta)$ and $((\alpha \in \delta_j \;\&\; \bar{\zeta} \in d_{j+k}) \vee (\alpha \in d_j \;\&\; \bar{\zeta} \in \delta_{j+k}))$. From $h_\ell \leqslant \underline{h}_\ell$, one has accordingly $((\alpha \in \underline{\delta}_j) \;\&\; \bar{\alpha} \in (d'_j \cup \delta'_j)) \vee (\alpha \in (\underline{d}_j \cup \underline{\delta}_j) \;\&\; \bar{\alpha} \in \delta'_j)$, a contradiction of $\underline{h}_j \# h'_j$.

Suppose $Ult(\rho'_i) \not\subseteq d''_i$ for some i. From $h'_i \in \mathcal{HC}$ and $h_\ell \leqslant \underline{h}_\ell$: $Ult(\rho'_i) \cap (\bigcup_{j=1}^{m-1} \pi^j(\delta_{j+i})) = \emptyset$. Hence, one can find some α in $Ult(\rho'_i)$ such that $\alpha \notin \bigcup_{j=1}^{m-1} \pi^j(d_{j+i})$. But $\alpha \in Ult(\rho'_i)$ implies $\alpha = \pi^k(\zeta)$ for some $k < m$ and $\zeta \in Ult(\rho_{i+k})$, and the results a contradiction of $h_{i+k} \in Q \in \mathcal{HC}$.

§ We show $h_i \# h''_i$ for every $i > 0$:

$(\forall \ell)(h_\ell \leqslant \underline{h}_\ell) \Rightarrow h''_i \leqslant h'_i$,

$(h_i \leqslant \underline{h}_i \;\&\; h''_i \leqslant h'_i \;\&\; \underline{h}_i \# h'_i) \Rightarrow h_i \# h''_i$.

§ From the above facts, the set $\lambda : [\![\pi]\!](h_{i+1} \Vdash h''_{i+1})$ is well defined, and $h''_i \in \lambda : [\![\pi]\!](h_{i+1} \Vdash h''_{i+1})$ for every i, whence clearly $h''_0 \in G'(Q)$. To end with, we shall prove $h'_0 \in norm(h''_0)$ from relations $\delta'_0 = \bigcup_{j=1}^{m-1} \pi^j(\underline{\delta}_j)$, $d'_0 = (\bigcup_{j=1}^{m-1} \pi^j(\underline{d}_j)) \setminus \delta'_0$ and $\langle \underline{\delta}_i, \rho_i, \underline{d}_i \rangle \in norm(\langle \delta_i, \rho_i, d_i \rangle)$. We are clearly free to assume $(\forall i \geqslant m)(\underline{\delta}_i = \emptyset = \underline{d}_i)$.

Under this assumption, $norm(h''_m) = \{h''_m\} = \{h'_m\}$; by the induction on n, the desired result is now a straightforward consequence of the identity :

$norm(\lambda : [\![\pi]\!](h' \| h'')) = (\lambda : [\![\pi]\!](norm(h') \| norm(h")))$

<u>proof of lemma 4.2</u> It is enough to consider the simplified situation where h', h'' belong to $\mathcal{H}_L$ for L equal to $\{\alpha, \bar{\alpha}\}$. In the sequel, $h \in \mathcal{H}_L$ is said <u>normal</u> iff $norm(h) = \{h\}$. We proceed by case analysis.

§ <u>h' and h'' are normal</u>: the result is immediate.

§ <u>neither h' nor h'' is normal</u>. It is enough to deal with cases 1 and 2 below.

<u>case 1</u>. $h' = \langle \emptyset, \rho', \alpha \rangle$, $h'' = \langle \emptyset, \rho'', \bar{\alpha} \rangle$.
Then, there exists a single pair of compatible histories taken from $norm(h')$ resp. $norm(h'')$, namely $(\langle \emptyset, \rho', \alpha\bar{\alpha} \rangle, \langle \emptyset, \rho'', \alpha\bar{\alpha} \rangle)$, whence the result.

<u>case 2</u>. $h' = \langle \emptyset, \rho', \alpha \rangle$, $h'' = \langle \emptyset, \rho'', \alpha \rangle$.
We distinguish between subcases $\alpha \in Ult(\rho')$ - or $\alpha \in Ult(\rho'')$ - and $\alpha \notin (Ult(\rho') \cup Ult(\rho''))$.

<u>$\alpha \in Ult(\rho')$</u>. Since $\alpha \in Ult(\rho)$ for every ρ in $\rho' \| \rho''$, $norm(h' \| h'') = \langle \emptyset, \rho' \| \rho'', \alpha\bar{\alpha} \rangle$. Now, the single pair of compatible histories taken from

norm(h') resp. norm(h") is again the pair $(\langle \emptyset, \rho', \alpha\bar{\alpha} \rangle, \langle \emptyset, \rho'', \alpha\bar{\alpha} \rangle)$.

$\alpha \notin Ult(\rho') \;\&\; \alpha \notin Ult(\rho'')$. Certainly, $\alpha \notin Ult(\rho)$ for $\rho \in \rho' \| \rho''$, and norm$(h' \| h'')$ is therefore equal to $(\langle \alpha, \rho' \| \rho'', \emptyset \rangle + \langle \emptyset, \rho' \| \rho'', \alpha\bar{\alpha} \rangle)$. Accordingly, one observes exactly two pairs of compatible histories taken from norm(h') resp. norm(h"), namely $(\langle \alpha, \rho', \emptyset \rangle, \langle \alpha, \rho'', \emptyset \rangle)$ and $(\langle \emptyset, \rho', \alpha\bar{\alpha} \rangle, \langle \emptyset, \rho'', \alpha\bar{\alpha} \rangle)$.

§ $h' = \langle \emptyset, \rho', \alpha \rangle$ and h" is normal. We deal separately with situations $\alpha \in Ult(\rho')$ and $\alpha \notin Ult(\rho')$.

case 1. $\alpha \in Ult(\rho')$.

Then norm$(h') = \langle \emptyset, \rho', \alpha\bar{\alpha} \rangle$. Set $h'' = \langle \delta'', \rho'', d'' \rangle$, possible subcases are as follows.

$\alpha \in \delta'' \cup \bar{\delta}''$

Then $h' \| h'' = \emptyset$ = norm$(h') \|$ norm(h'').

$\alpha \notin \delta'' \cup \bar{\delta}'' \cup Ult(\bar{\rho}'')$

Since $\alpha \notin \delta'' \cup \bar{\delta}''$, h' and h" are compatible histories and $h' \| h'' = \langle \emptyset, \rho' \| \rho'', d'' \cup \{\alpha\} \rangle$. But α belongs to $Ult(\rho)$ for every ρ in $\rho' \| \rho''$, and thus norm$(h' \| h'')$ is equal to $\langle \emptyset, \rho' \| \rho'', \alpha\bar{\alpha} \rangle$. The result is clear from $\langle \emptyset, \rho', \alpha\bar{\alpha} \rangle \# \langle \emptyset, \rho'', d'' \rangle$.

$\alpha \notin \delta'' \cup \bar{\delta}''$, $\bar{\alpha} \in Ult(\rho'')$.

Obvious from the identity $h'' = \langle \emptyset, \rho'', \alpha\bar{\alpha} \rangle$.

case 2 . $\alpha \notin Ult(\rho')$.

Then $norm(h') = \{<\alpha, \rho', \emptyset>, <\emptyset, \rho', \alpha\bar{\alpha}>\}$.

Set $h'' = <\delta'', \rho'', d''>$, possible subcases are as follows.

$\neg(h' \# h'')$

Then $\bar{\alpha} \in \delta''$ and $norm(h') \| h'' = \emptyset$.

$h' \# h''$

Since h'' is normal, either a, b or c must be found.

a) $h'' = <\emptyset, \rho'', \emptyset>$: then $\forall \rho \in \rho' \| \rho''$, $\alpha \notin Ult(\rho)$, and $norm(h' \| h'')$ amounts therefore to the set $<\alpha, \rho' \| \rho'', \emptyset> + <\emptyset, \rho' \| \rho'', \alpha\bar{\alpha}>$.

b) $h'' = <\emptyset, \rho'', \alpha\bar{\alpha}>$: then $<\alpha, \rho', \emptyset>$ and h'' are not compatible, whence $norm(h') \| norm(h'')$ amounts to $<\emptyset, \rho' \| \rho'', \alpha\bar{\alpha}>$.

c) $h'' = <\alpha, \rho'', \emptyset>$: then $<\emptyset, \rho', \alpha\bar{\alpha}>$ and h'' are not compatible, whence $norm(h') \| norm(h'')$ amounts to $<\alpha, \rho' \| \rho'', \emptyset>$.

end of proofs

5. REMARKS ON THE RATIONAL SUBSET.

definition 5.1. The rational subset $T_{\Sigma'}$ of T_Σ is the term algebra on Σ', the signature obtained from Σ when removing the network iterators $[\lambda \triangleright \pi]$.

An important property of $T_{\Sigma'}$ is to generate, for every Σ-program p, the entire set $OBS(p)$ of p's observations, as we state formally in the ...

proposition 5.1. $\forall p \in T_\Sigma$, $\forall h \in OBS(p)$, $\exists q \in T_{\Sigma'}$, $\exists \mathcal{C} \in fair(p|q) : h = abs(\mathcal{C}_\rightarrow)$.

proof The arguments which have been used to prove proposition 3.2 may be used as well to establish $\mathcal{H} = \varphi(trace(FAIR(T_{\Sigma'})))$. The property to be demonstrated is then clear from proposition 2.3, definition 3.1 and proposition 3.1 □

Let OBS' and $\lesssim'$ denote the respective transpositions of OBS and $\lesssim$ to $T_{\Sigma'}$, i.e. $OBS'(p) = \{ abs(\mathcal{C}_\rightarrow) \mid \mathcal{C} \in fair(p|q) \text{ for some } q \in T_{\Sigma'} \}$ and $p \lesssim' q$ iff $OBS'(p) \subseteq OBS'(q)$. If $\mathcal{N}'$ is the restriction of $\mathcal{N}$ to $T_{\Sigma'}$, then we can state without proof the following variant of theorem 4.1 :

theorem 5.1 $\mathcal{N}'$ is a fully abstract model of $T_{\Sigma'}$ w.r.t. the observational preorder $\lesssim'$:

$\mathcal{N}'(t) \sqsubset_I \mathcal{N}'(t')$ iff for every Σ'-context $\mathcal{C}'$, $OBS'(\mathcal{C}'(t)) \subseteq OBS'(\mathcal{C}'(t'))$. Moreover, the following equivalence holds: $\mathcal{N}'(t) \sqsubset_I \mathcal{N}'(t')$ iff $OBS'(t) \subseteq OBS'(t')$ □

The rational subset draws its practical interest from the following ...

<u>theorem 5.2</u> There exists an effective procedure for deciding $\mathcal{N}'(p) \sqsubset_I \mathcal{N}'(q)$.

<u>proof</u>. Let $Rat(\Delta^\infty)$ denote the set of the infinitary rational languages over the finite subsets of Δ, and let $\|$ denote the extension to languages of the operation defined in 3.7, then $\|$ is an internal operation on $Rat(\Delta^\infty)$, and this operation is effective [Si] [DK2]. Hence, from the finiteness of program sorts and by the induction on Σ'-terms, $\mathcal{N}'(p)$ may be computed in the form of a finite sum $\sum_i \langle \delta_i, \mathcal{L}_i, d_i \rangle$ where the $\mathcal{L}_i$ are in $Rat(\Delta^\infty)$. Now, the inclusion is a decidable property for $Rat(\Delta^\infty)$ - see for instance [Ei] □

A weaker form of this theorem was the central result of [DK1], where the rational subset was given a more attractive syntax with recursive style declarations. Both open and closed terms

were provided in that syntax, say S. The open terms are constructed from variables x_i and from closed terms by the exclusive use of the n-ary guarding operators. NIL is a closed term. The other closed terms are constructed from closed terms by the application of the n-ary guarding, parallel composition and renaming operators, or they are generated from open terms by the use of the recursion contrinator Rec, e.g. in $Rec(x_1 \leftarrow t_1, \dots, x_n \leftarrow t_n)$.

Programs p written in S may be translated into CS according to the following process :

a - as long as there occurs in p some term $t = Rec(x_1 \leftarrow t_1, \dots, x_n \leftarrow t_n)$ such that some of the t_i, say t_1, has form $\langle \lambda_1, \dots, \lambda_m \rangle (u_1, \dots, u_m)$ where some of the u_j is neither a variable nor a closed term, replace t by t' as follows :
$t' = Rec(x_1 \leftarrow \langle \lambda_1, \dots, \lambda_m \rangle (x_{n+1}, \dots, x_{n+m}), x_2 \leftarrow t_2, \dots, x_n \leftarrow t_n, x_{n+1} \leftarrow u_1, \dots, x_{n+m} \leftarrow u_m)$;

b - as long as there occurs in p some recursive definition $D = Rec(x_1 \leftarrow t_1, \dots, x_n \leftarrow t_n)$, chose fresh identifiers $\alpha_1, \dots, \alpha_n$ in Λ and replace D by
$D' = ((\langle \overline{\alpha}_1 \rangle (NIL) \mid P)[\varphi])[\psi]$
where :

- φ and ψ are renaming functions such that φ is

undefined for the α_i and $\psi \circ \varphi$ acts as the identity on $\bigcup_i sort(t_i)$,

- $P = (P_1 \mid (P_2 \mid \ldots P_n))$ with

$P_i = \langle \alpha_i \rangle (t_i)$ if t_i is a closed term, or else

$P_i = \langle \alpha_i \Rightarrow \lambda_{i_1} : \bar{\alpha}_{j_1}, \ldots, \lambda_{i_n} : \bar{\alpha}_{j_n} \rangle$ for t_i equal to $\langle \lambda_{i_1}, \ldots, \lambda_{i_n} \rangle (x_{j_1}, \ldots, x_{j_n})$.

The properties established in [DK2] for linear systems upon $Rat(Z^\infty)$ show that the S-semantics of p is preserved by transformation $\bar{a}$. Besides, the operators NIL, $\|$ and $[\varphi]$ are given equivalent semantics in S and CS. A careful comparison of the computations of the above programs D and D' may be used to prove that the semantic equivalence between S and CS programs is universally valid through the proposed translation. The heart of the proof is to demonstrate that for any L-sorted triple $\langle \delta, \rho, d \rangle$ satisfying

$((d \cap \delta = \emptyset) \& (\bar{d} \subseteq d \cup \delta) \& (Ult(\rho) \subseteq d))$,

if Θ denotes the transformation given by

$\langle \delta, \rho, d \rangle \xrightarrow{\Theta} \langle (L \cup \bar{L}) \setminus (d \cup \delta), \rho, d \rangle$,

then $A \xleftrightarrow{\Theta} B$ holds for sets A and B :

$A = \wedge \{ h \in \mathcal{H}_L \mid h \preccurlyeq \langle \delta, \rho, d \rangle \}$,

$B = \vee \circ norm \circ \Theta (\langle \delta, \rho, d \rangle)$.

6. SOME CONCLUSIONS.

The fair composition of infinitary words must be accounted for some way or other in any kind of model of fair asynchrony : we do not see the reason why these models should not be constructed around that central concept, and thus in the framework of infinitary languages. In this paper, we have shown that fully observational models may be obtained this way for communicating programs. It was clear from a long time that conventional traces do not provide an adequate semantics of programs : beside the effective trace, some indication must be given about the alternative actions which are avoided. This is why our "history items", or elementary action descriptors, are triples whose middle element is an effective action. Histories $\langle \delta, \rho, d \rangle$ are extracted from finite or infinite sequences of action records by applying morphisms tailored down to gather exactly what is needed. Here, the side elements δ and d are extracted according to different criteria which reflect weak fairness resp. strong fairness. The fact that our model $\mathcal{N}$ has been proven fully abstract w.r.t. the most discriminative observational congruence indicates

without any doubt that branching time logic is inadequate to reason about observable behaviours. In our sense, programs p and q are observationally equivalent iff for every r, the set of the possible computations of r in $p|r$ is identical to the set of the possible computations of r in $q|r$. We conjecture that the previous equivalence is strictly included into the testing equivalence of Hennessy and de Nicola, but no difference appears for the rational subset. More sensible than the choice between finite and infinite observations is the choice between synchrony and asynchrony. Compared to Hennessy's study of fair delay operators for the synchronous case, our work follows a parallel way for the asynchronous case. But we feel that no junction exists between the two ways. Clearly, synchrony cannot be simulated by asynchrony. In the other direction, an observation congruence suited to fair asynchrony may perhaps be derived from a "synchronous" congruence, but the derived relation cannot coincide with the "asynchronous" congruence which is obtained when considering asynchronous observers only.

REFERENCES

[AB] Austry D., Boudol G. *Algèbre de processus et synchronisation.*
TCS 30, 1 (pp. 91-131) -1984-

[Br-1] Brookes S., *On the relationship of CCS and CSP.*
Proc. ICALP 83, LNCS 154 -1983-

[Br-2] Brookes S., *A Model for Communicating Sequential Processes.*
Ph. D. thesis, University of Oxford -1983-

[CS] Costa G., Stirling C., *A Fair Calculus of Communicating Systems.*
Proc. FCT 83, LNCS 158 -1983-

[Da] Darondeau Ph., *An enlarged definition and complete axiomatization of Observational Congruence of Finite Processes.*
Proc. 5th Int. Symp. on Prog., LNCS 137 -1982-

[DK] Darondeau Ph., Kott L., *On the observational Semantics of Fair Parallelism.*
1) Proc. ICALP 83, LNCS 154 -1983-
2) INRIA report 262 -1983-

[Ei] Eilenberg S., *Automata Languages and Machines*, Vol. A, Academic Press -1974-

[HBR] Hoare C., Brookes S., Roscoe A., *A Theory of Communicating Sequential Processes.* Report PRG 16, University of Oxford - 1981 -

[He-1] Hennessy M. *A Model of Nondeterministic Machines.* Report CSR 135-83, University of Edinburgh - 1983.

[He-2] Hennessy M. *Modelling Finite Delay Operators* Report CSR 153-83, University of Edinburgh - 1983 -

[HM] Hennessy M., Milner R., *On observing non-determinism and Concurrency* Proc. ICALP 80, LNCS 85 - 1980 -

[HP] Hennessy M., Plotkin G., *A term model for CCS* Proc. 9th MFCS, LNCS 88 - 1980 -

[HN] Hennessy M, de Nicola R., *Testing equivalences for Processes.* Proc. ICALP 83, LNCS 154 - 1983 -

[Jo] Jorrand Ph., *Specification of Communicating Processes and Process Implementation Correctness* Proc. 5th Int. Symp. on Prog., LNCS 137 - 1982 -

[K] Kennaway J., *Formal Semantics of nondeterminism and parallelism* Ph.D. thesis, University of Oxford - 1981 -

[Mi-1] Milner R., *A Calculus of Communicating Systems.* LNCS 92 - 1980 -

[Mi 2] Milner R., Calculi for Synchrony and Asynchrony
TCS 25, 3 (pp. 267-310) -1983-

[Mi 3] Milner R., Fully Abstract Models of Typed Lambda-Calculi
TCS 4, 1 (pp. 1-23) -1977-

[Mi 4] Milner R., A Finite Delay Operator for Synchronous CCS
Report CSR 116-82, University of Edinburgh -1982-

[Si] de Simone R., Langages Infinitaires et Produits de Mixage
to appear in TCS.

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique